플러터를 이용한 앱 개발

Design Thinking, Figma, Git, Dart, Flutter

WORKBOOK

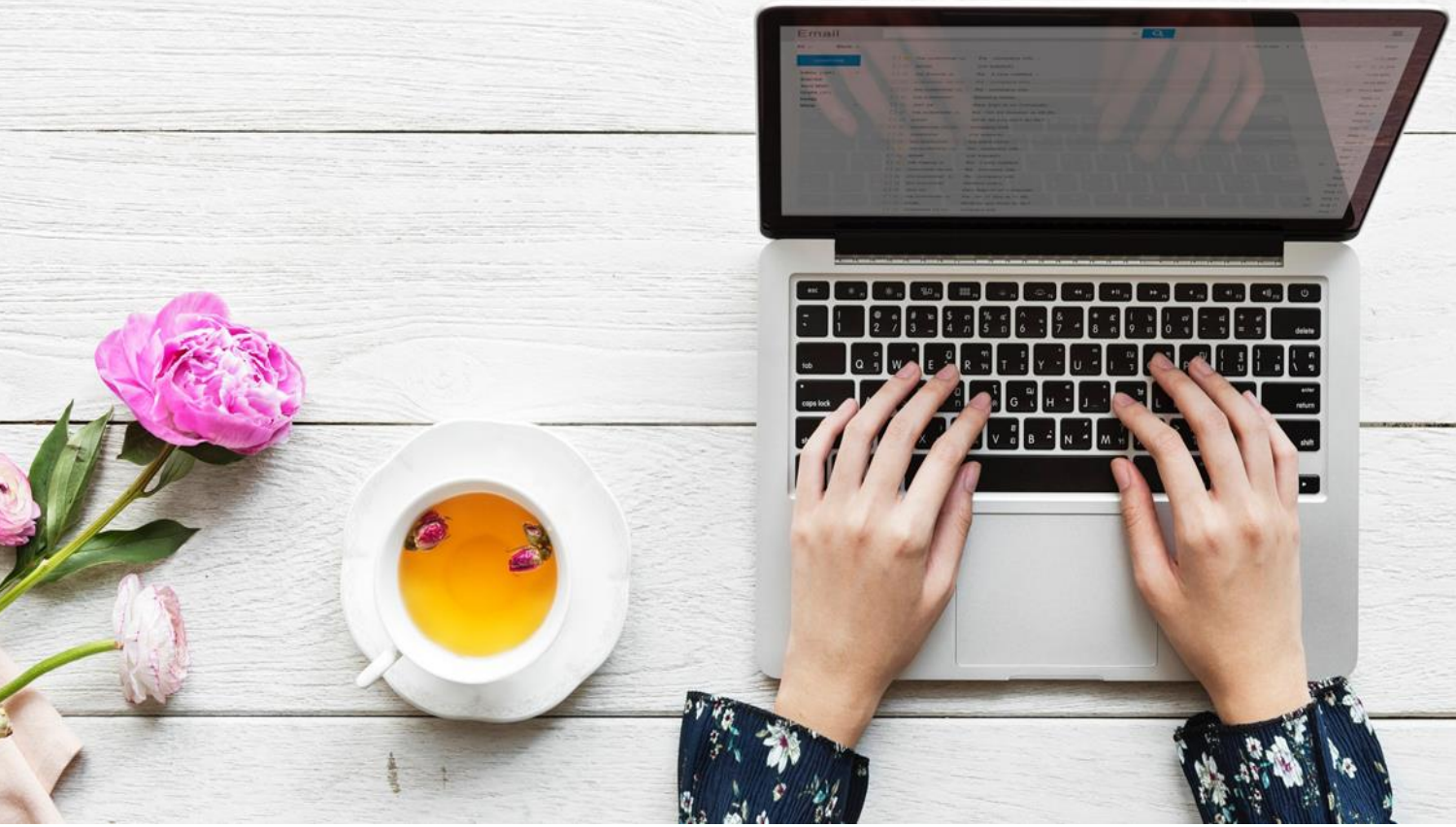
한신대학교 SW 교육센터

박종현 교수

충남대학교 소프트웨어중심대학사업단

김도헌 산학협력중점교수

충남대학교 소프트웨어중심대학사업단
Chungnam National University National Center of Excellence in Software



본 교재는 2023 년도 과학기술정보통신부 및 정보통신기획평가원의 'SW 중심대학사업' 지원을 받아 제작되었습니다.

플러터를 이용한 앱 개발

- Design Thinking, Figma, Figma, Dart, Flutter workbook

발행 2023 년 11 월 30 일

지은이 김도헌, 박종현

발행인 -

발행처 -

전자우편 – 김도헌(artcoding@cnu.ac.kr)

홈페이지 -

ISBN - 978-89-97590-80-3

목차

기초프로젝트랩 워크북

기초프로젝트랩 교과목에서는 Design Thinking 을 바탕으로 모바일 앱 프로그래밍 프로젝트를 진행합니다.

본 워크북은 학생 여러분이 프로젝트를 진행할 때 알아야 할 기본 기술 분야를 요약하여 프로젝트 진행을 원활히 할 수 있도록 하기 위해 제작되었습니다. 수업 시간에 활용되며 이론 강의와 실습으로 이루어져 있습니다.

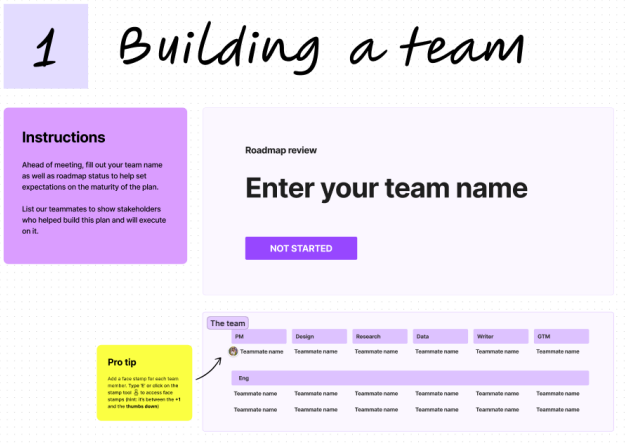
팀 구성하기

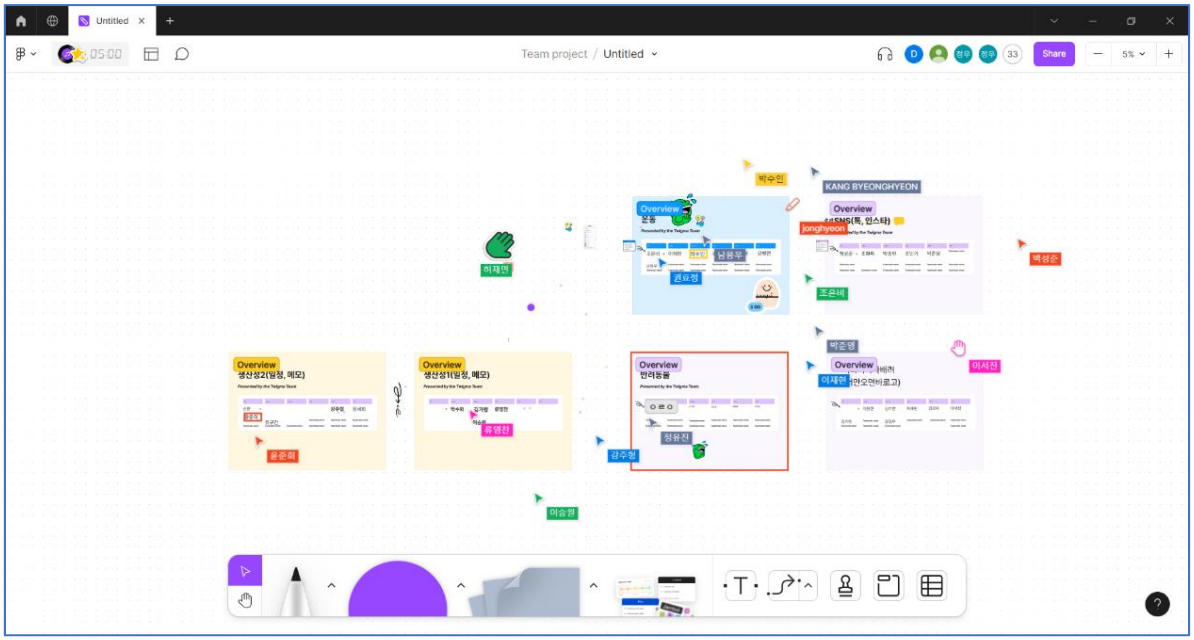
FigJam 을 통해서 주제별로 팀 구성하기 (6±1 명)

FigJam Template : Product roadmap review

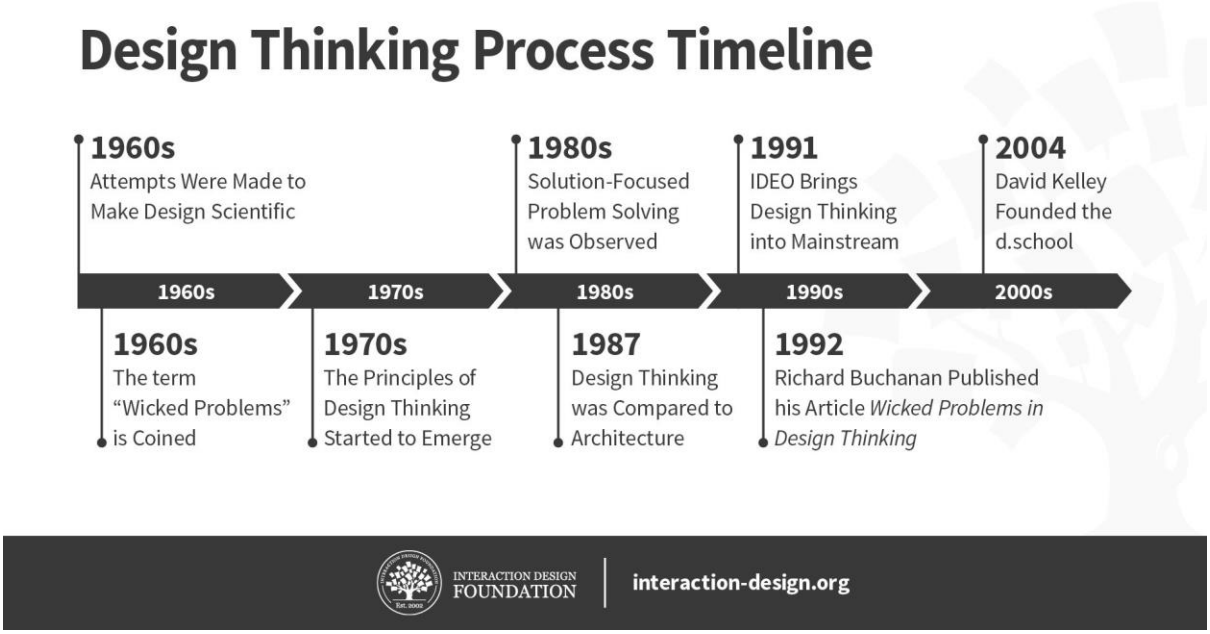
FigJam Share Link 를 Slack 통해서 공유 후 실시간으로 팀 구성

주제 예시)　건강, 운동, SNS(사진, 톡 등), 생산성(일정, 메모 등), 사회적약자배려, 반려동물
필요한 주제는 추가할 수 있음





1. Design Thinking



https://www.ideou.com/pages/design-thinking

https://dschool.stanford.edu/

https://www.interaction-design.org/literature/topics/design-thinking

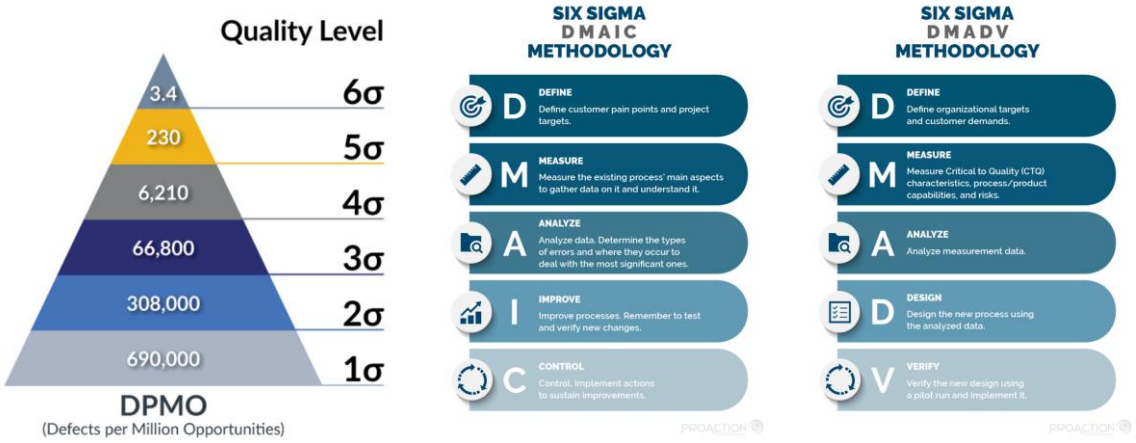
Design thinking is a non-linear, iterative process that teams use to understand users, challenge assumptions, redefine problems and create innovative solutions to prototype and test. Involving five phases—Empathize, Define, Ideate, Prototype and Test—it is most useful to tackle problems that are ill-defined or unknown.

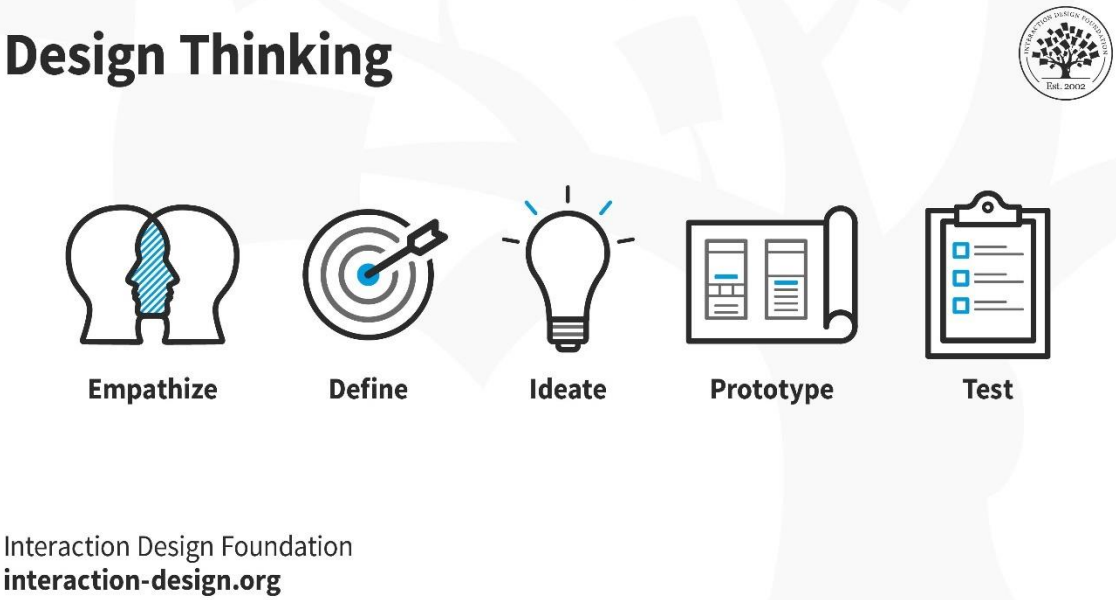
APPENDIX : six-sigma

https://ko.wikipedia.org/wiki/6_%EC%8B%9C%EA%B7%B8%EB%A7%88
https://blog.naver.com/wjn21/221010380254
https://blog.proactioninternational.com/en/six-sigma

6 시그마(6σ)는 기업에서 전략적으로 완벽에 가까운 제품이나 서비스를 개발하고 제공하려는 목적으로 정립된 품질 경영 기법 또는 철학으로서, 기업 또는 조직 내의 다양한 문제를 구체적으로 정의하고 현재 수준을 계량화하고 평가한 다음 개선하고 이를 유지 관리하는 경영 기법이다. 원래 모토로라에서 개발된 일련의 품질 개선 방법이었으며 품질 불량의 원인을 찾아 해결해 내고자 하는 체계적인 방법론이었다.

이후 제너럴 일렉트릭 등 여러 기업에서 도입되어 발전하였으며 특히 1990 년대와 2000 년대 동안 많은 인기를 얻은 기업 내 혁신을 위한 방법이다. 다른 품질경영관리기법인 종합 품질 관리(Total Quality Management)의 경우에는 생산 품질 자체에 집중하지만 6 시그마는 회사의 모든 부서의 업무에 적용할 수 있으며 각자의 상황에 알맞은, 고유한 방법론을 개발하고 적용하여 정량적 기법과 통계학적 기법으로 향상시킬 수 있다.





1-1. Empathize

: 사용자의 요구 사항 조사

일반적으로 사용자 조사를 통해 해결하려는 문제에 대한 공감적인 이해를 얻어야 합니다. 공감은 세상에 대한 자신의 가정을 제쳐 두고 사용자와 그들의 요구에 대한 진정한 통찰력을 얻을 수 있게 해주기 때문에 디자인 사고와 같은 인간 중심의 디자인 프로세스에 매우 중요합니다.

1-2. Define

: 사용자의 요구 사항과 문제 명시

공감 단계에서 수집된 정보를 모아보는 시간입니다. 그런 다음 관찰 내용을 분석하고 종합하여 팀이 식별한 핵심 문제를 정의합니다. 이러한 정의를 "problem statements" 이라고 합니다 . 아이디어 구상을 진행하기 전에 인간 중심의 노력을 유지하는데 도움이 되는 페르소나를 만들 수 있습니다.

Personas

https://youtube.com/watch?v=zOQp7ElflSI&feature=shared

1-3. Ideate

: 가정에 도전하고 아이디어 창출

처음 두 단계의 탄탄한 지식 배경은 "틀 밖에서 생각하기" 시작할 수 있고, 문제를 보는 대안적인 방법을 찾고, 작성한 문제 설명에 대한 혁신적인 솔루션을 식별할 수 있음을 의미합니다. 브레인스토밍은 여기서 특히 유용합니다.

1-4. Prototype

: 솔루션 생성 시작

이는 실험적인 단계입니다. 목표는 발견된 각 문제에 대해 가능한 최상의 솔루션을 식별하는 것입니다. 팀은 생성한 아이디어를 조사하기 위해 저렴하고 축소된 제품 버전(또는 제품 내 특정 기능)을 생산해야 합니다. 여기에는 단순히 종이 프로토타입 제작이 포함될 수 있습니다.

1-5. Test

: 솔루션을 시험해 보세요

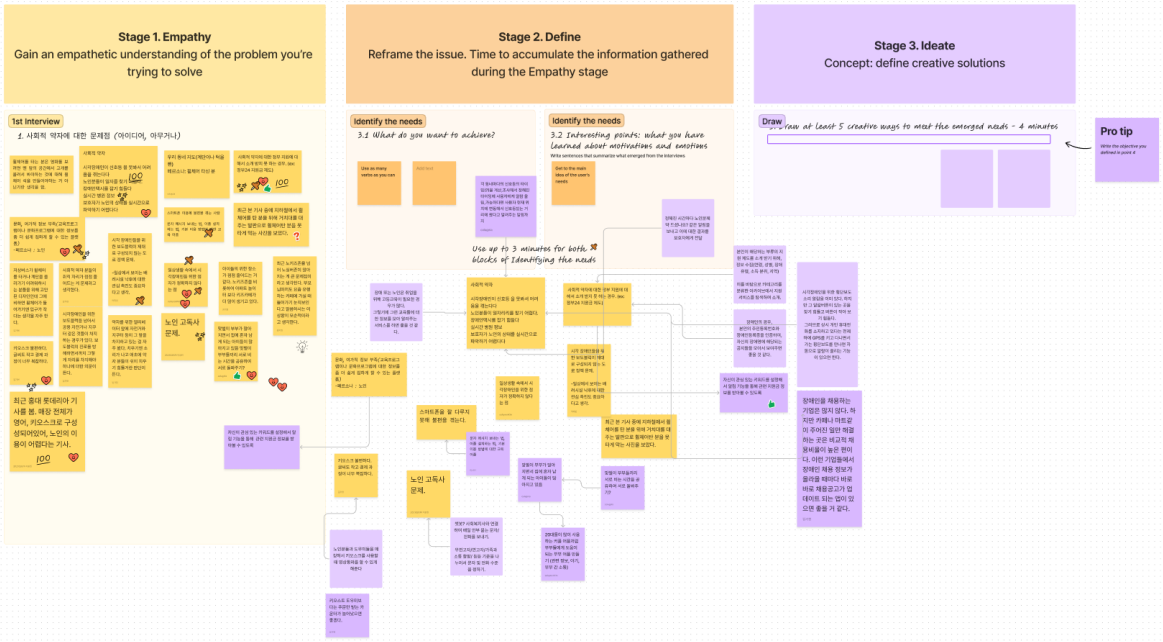
평가자는 프로토타입을 엄격하게 테스트합니다. 이것이 마지막 단계이기는 하지만 디자인 사고는 반복적입니다. 팀은 종종 결과를 사용하여 하나 이상의 추가 문제를 재정의합니다. 따라서 이전 단계로 돌아가 추가 반복, 변경 및 개선을 수행하고 대체 솔루션을 찾거나 배제할 수 있습니다.

관련 영상

https://youtube.com/watch?v=GDz9tIfyBFk&feature=shared
https://youtube.com/watch?v=ILCnD61jdwk&feature=shared
https://youtube.com/watch?v=0J3476ZXmTM&feature=shared

[실습]

사회적 약자 배려 (6조)



2. Figma

참고문헌) Do it! 프로덕트 디자인 입문 with 피그마 - 이지스퍼블리싱

https://www.figma.com/community/file/1172858272396686782/Do-it!-%ED%94%84%EB%A1%9C%EB%8D%95%ED%8A%B8-%EB%94%94%EC%9E%90%EC%9D%B8-%EC%9E%85%EB%AC%B8-with-%ED%94%BC%EA%B7%B8%EB%A7%88-(%EC%98%88%EC%A0%9C)

***** Figma 를 본격적으로 사용하기 전에 알아 두어야 할 기본 사항 *****

- Figma 는 웹 기반이며, 현재 제일 각광받는 디자인 툴이다.

설치가 필요 없고, OS 구별하지 않고, 공유 링크만으로 빠르게 협업이 가능하다.

https://uxtools.co/survey/2022/ui-design/#ui-design-tools-graph

- 피그마는 벡터(Vector)기반 툴 입니다.

벡터 : 점, 선, 면으로 이루어진 오브젝터, 위치와 색상값을 기억하므로 확대해도 깨지지 않는다. 확장자 AI, SVG, EPS

비트맵 : 확대하면 깨짐 발생

https://imweb.me/faq?mode=view&category=29&category2=33&idx=71515

- 피그마 플랜 : 스타터플랜(무료)

[Drafts] 공유하지 않는다면 페이지수나 파일 수 제한없이 생성과 편집 가능

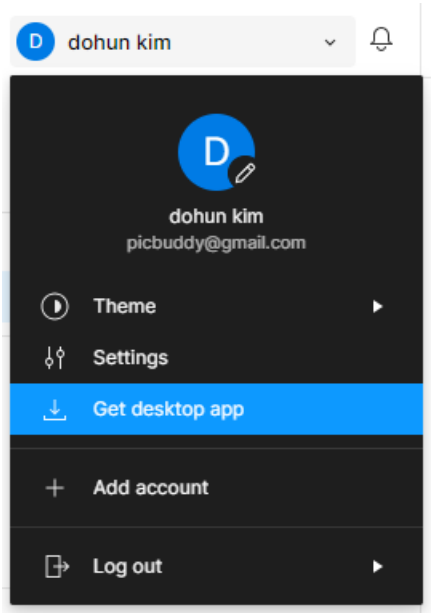
[Teams] 에디터(편집권한이 있는 작업자)가 2 명 이상일 때, 파일 최대 3 개 (한 파일당 페이지수 3 개 이하)

https://www.figma.com/pricing/?fuid=1195581506274032374#cid-20p4Q0lWSOxIWj30zTY6P2

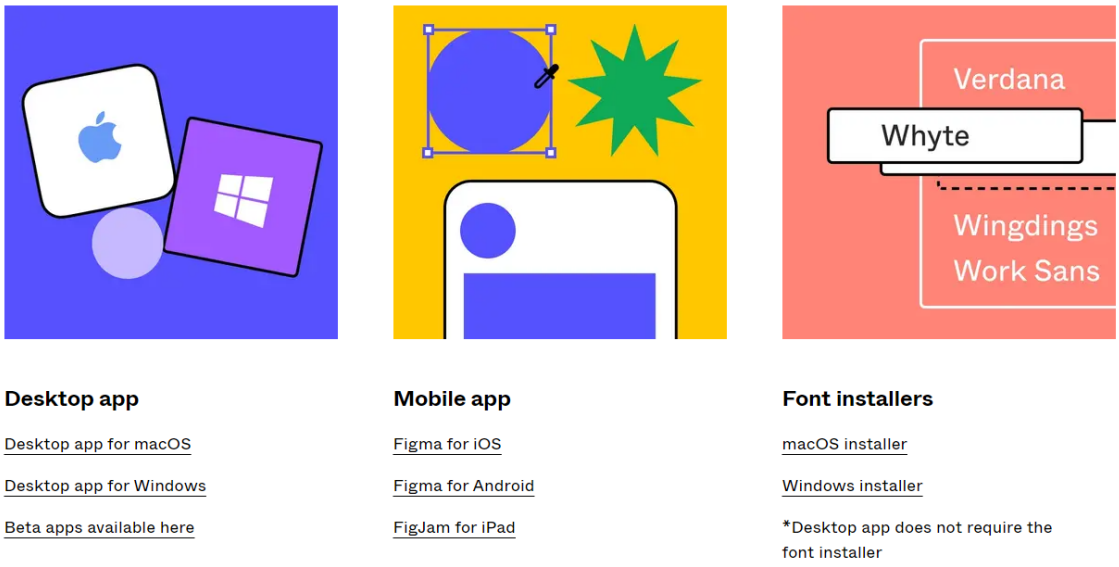
- 교육용 계정으로 신청할 수 있습니다.

https://www.figma.com/education/
https://growingsaja.tistory.com/654
https://edutech-lab.tistory.com/32

- 웹브라우저 이외에도 데스크톱용 앱(클라이언트), 모바일 앱 설치 가능함.



Figma downloads



Desktop app

Desktop app for macOS

Desktop app for Windows

Beta apps available here

Mobile app

Figma for iOS

Figma for Android

FigJam for iPad

Font installers

macOS installer

Windows installer

*Desktop app does not require the font installer

https://www.figma.com/downloads/

- 모바일앱 "미러" 기능 사용해 보기

1. 앱스토어 또는 구글플레이에서 "Figma"를 검색하여 앱 설치
2. 피그마 앱 로그인
3. 피그마 앱 하단의 "Mirror" 터치
4. 데스크톱 앱에서 프레임을 선택하면 모바일 앱에서 확인 가능
5. two finger 터치 → Exit

- 9 가지 기본 인터페이스

1. 피그마 로고 아이콘 : Main Menu
2. Move (v):

   Ctrl+클릭 : 가장 깊은 계층 선택

   Shift+모서리 클릭 : 가로 세로 비율 유지

Move 로 크기 조절하면 Constraints 규칙에 따라 요소가 자동 배치

Scale 로 크기 조절하면 Constraints 무시됨

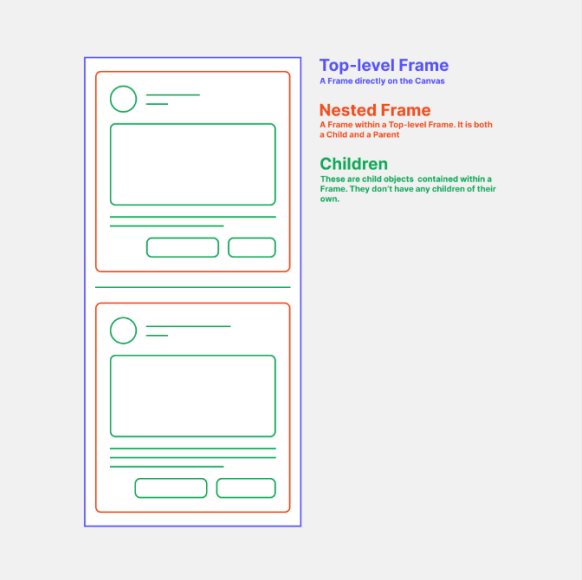
3. Region tools

3-1. Frame(F) → 마우스 커서가 "+"로 변경

드래그 하여 원하는 크기의 프레임 생성 or 프레임 크기 선택(Design 패널)

영역을 나누는 틀이다.

https://help.figma.com/hc/en-us/articles/360041539473-Frames-in-Figma



오브젝트를 먼저 만들고 전체를 프레임에 넣을 수도 있다 (오른쪽 클릭-Frame Selection, Ctrl+Alt+G)

3-2 Slice (S) : 스크린샷하듯 추출, 레이어 계층 무시하고 이미지 추출할 때 유용

4. Shape tools(사각형 R, 선 L, 화살표 Shift+L, 원 O)

5. Creation Tools : Pen(P), Pencil(Shift+P) : drawing tool

펜으로 외곽선 추출 후 Fill-Black, vector 레이어를 하위로 하고 반달 모양 아이콘으로 마스킹해보기

6. Text (T)

7. Resources (Shift+I)

Components : 직접 설정한 디자인 애셋 확인 및 사용

Plugins : 다양한 업무 효율화 툴

Widgets : 협업 상호 작용 기능

8. Hand tool(H)

캔버스 이동

오브젝트 선택, Shift+2 → Zoom to Selection

9. Add comment

@피그마 계정 : 코멘트 소통

참고문헌)

https://help.figma.com/hc/en-us

Figma for Beginners tutorial (4 parts)

Learn the basics by designing a social media app in Figma. We'll teach you about the fundamental tools you'll use, how to create wireframes, high-fidelity designs, and a prototype. We'll then go through the basics of developer handoff.

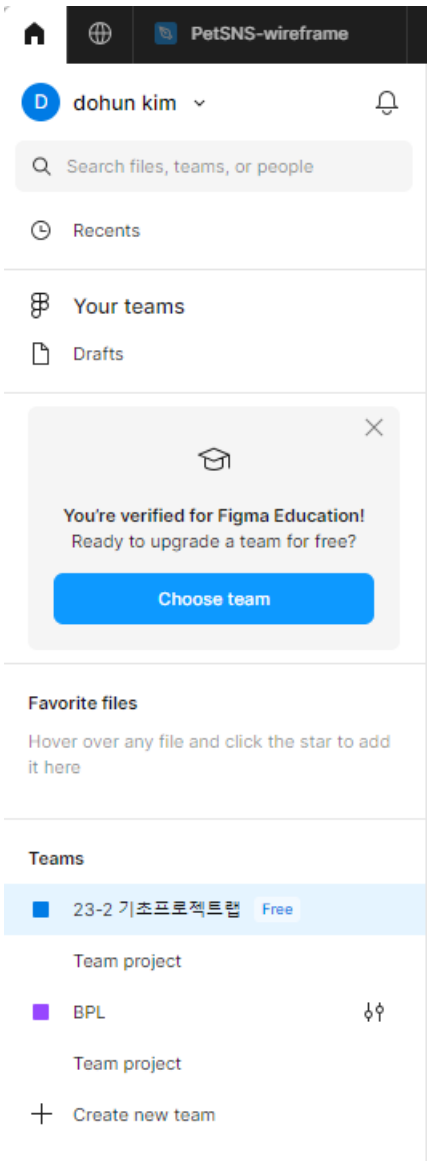
[실습]

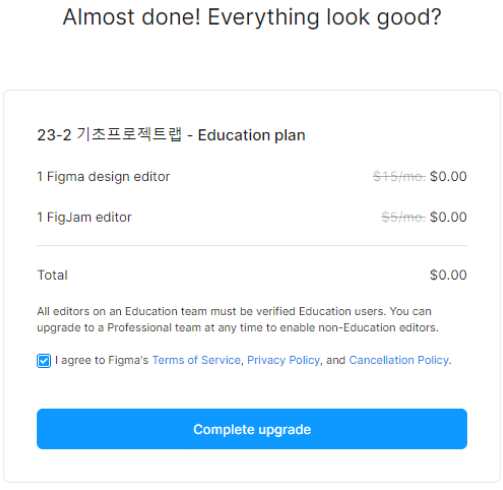
2-1. Wireframe

- 애완동물 소셜 미디어앱 아이디어 구체화

1. 와이어프레임 제작 : 색상과 같은 스타일이 없는 단순화된 디자인, 아이디어 피드백 용도

2. Create Team! → Team Project → +Design file → 캔버스 이름(petSNS), 팀 멤버초대





3. Layers , Page1 → Frame (f)- 모양, 이미지, 텍스트 컨테이너, 디자인의 단일 화면
4. 툴바 - 사이드바 - 특정(모델) 크기의 프레임(사전설정 : Frame→Archive→Google Pixel 2 XL)
5. 프레임 이름 변경(더블클릭 또는 오른쪽 클릭 후 "home" 으로 변경)
6. move, zoom, pan

   마우스휠 누르고 이동 or 스페이스바 누르고 이동

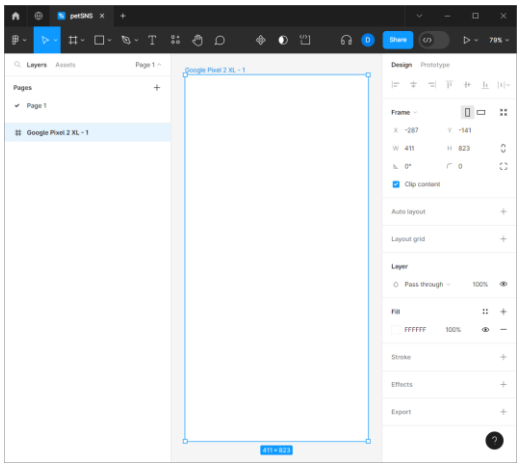
   줌인 줌아웃(cmd+-, ctrl+-, ctrl+스크롤)

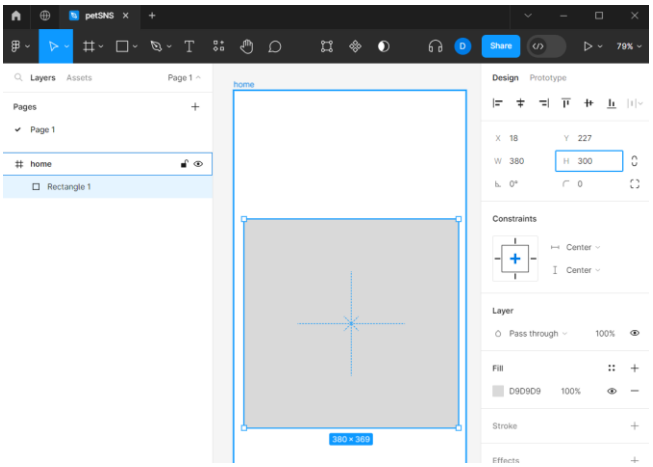
   zoom to fit (Shift + 1)

   보기 % 사용

7. Shape→Rectangle : w 380 x h 300 for pet image

   Rectangle 을 Frame 중앙으로 Align





8. Shape→Ellipse : Shift 키 사용 - 원으로 w 30 x h 30 for profile image
   사이드 바에서 화살표 키 사용 가능 (덧셈 같은 연산 가능+10) 40 크기로 변경
9. fill : 두 레이어 선택 - 회색 - 색상, 그라디언트, 이미지 선택 가능, "-버튼"으로 투명하게
10. Stroke : 와이어프레임이므로 "+버튼" 눌러서 검정 외곽선으로 변경
11. Rectangle 기준으로 Circle 을 왼쪽 정렬함.(더 큰 이미지 기준으로 정렬됨)
    Text : Name 입력 - 두 레이어 선택 후 Vertical Align

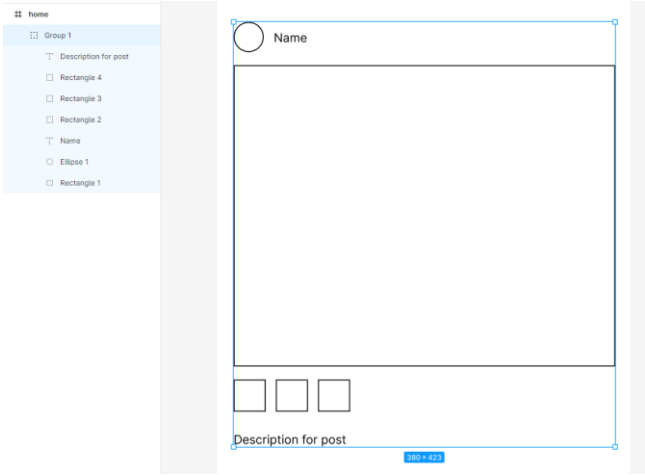
    화살표키로 위치 이동 할 수 있음 (alt 키 누르면 간격 숫자 볼 수 있음)

12. Placeholder icons : 댓글, 게시물 저장, 참여
    R 사각형 도구 - shift 정사각형 - 32x32 크기 - fill 빼기 - Stroke 검정

레이어 복사(Alt 또는 option 누르고 drag → cmd D/ctrl D)

→ 3 개 사각형 선택 후 간격 업데이트 10

13. "Description for post" text 추가
14. Group layers (layer 선택 후 오른쪽마우스 group selection) : 그룹화(cmd g/ctrl G)
15. Duplicate (alt/option + drag) : frame 밖으로 보이지 않으면 clip content 속성 해제

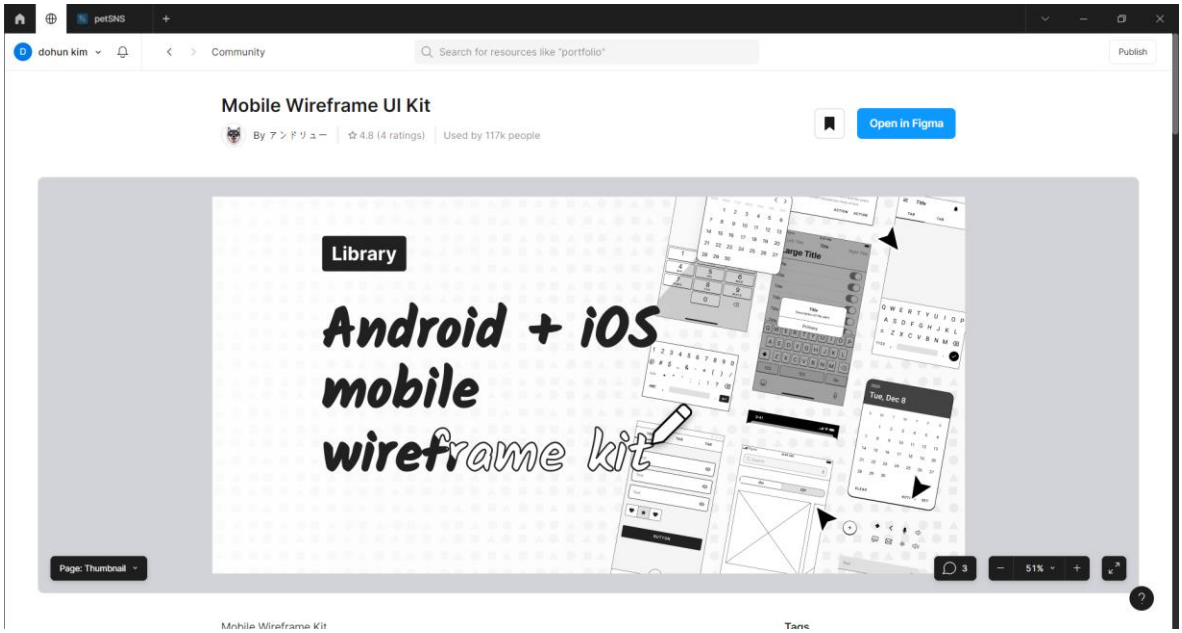


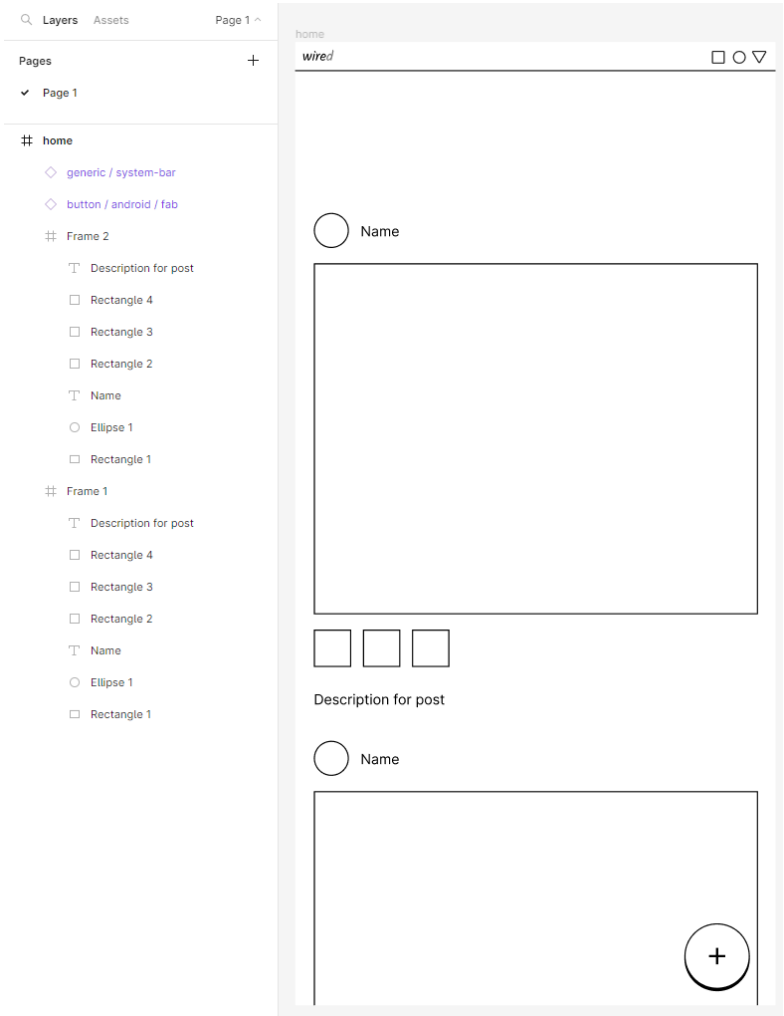
Components : 버튼, 아이콘, 도구모음, 메뉴 등

Libraries : components & styles

Design system : Components + Libraries + 코드로 구현하기 위한 지침

1. 커뮤니티 라이브러리 이용 (mobile wireframe UI Kit)
   open in Figma → move to project → publish library
   Asset → search →add to file
2. Android / App bars and navigation / system-bar → Drag → Frame 맞춤 (세로선 끌기)
3. Android / Buttons / button / android / fab-mini → Drag → 우 하단에 배치


petSNS
dohun kim
Community
Search for resources like "portfolio"
Publish
Mobile Wireframe UI Kit
By アンドリュー
4.8 (4 ratings)
Used by 117k people
Open in Figma
Library
Android + iOS
mobile
wireframe kit
Page: Thumbnail
51%
Mobile Wireframe Kit
Tags


Layers
Assets
Page 1
Pages
Page 1
home
generic / system-bar
button / android / fab
Frame 2
Description for post
Rectangle 4
Rectangle 3
Rectangle 2
Name
Ellipse 1
Rectangle 1
Frame 1
Description for post
Rectangle 4
Rectangle 3
Rectangle 2
Name
Ellipse 1
Rectangle 1
home
wired
Name
Description for post
Name

4. Constraints (프레임 크기 변경에 종속적이지 않는 문제 해결)

   Status bar : Left&Right&Top

   Fab : right & bottom

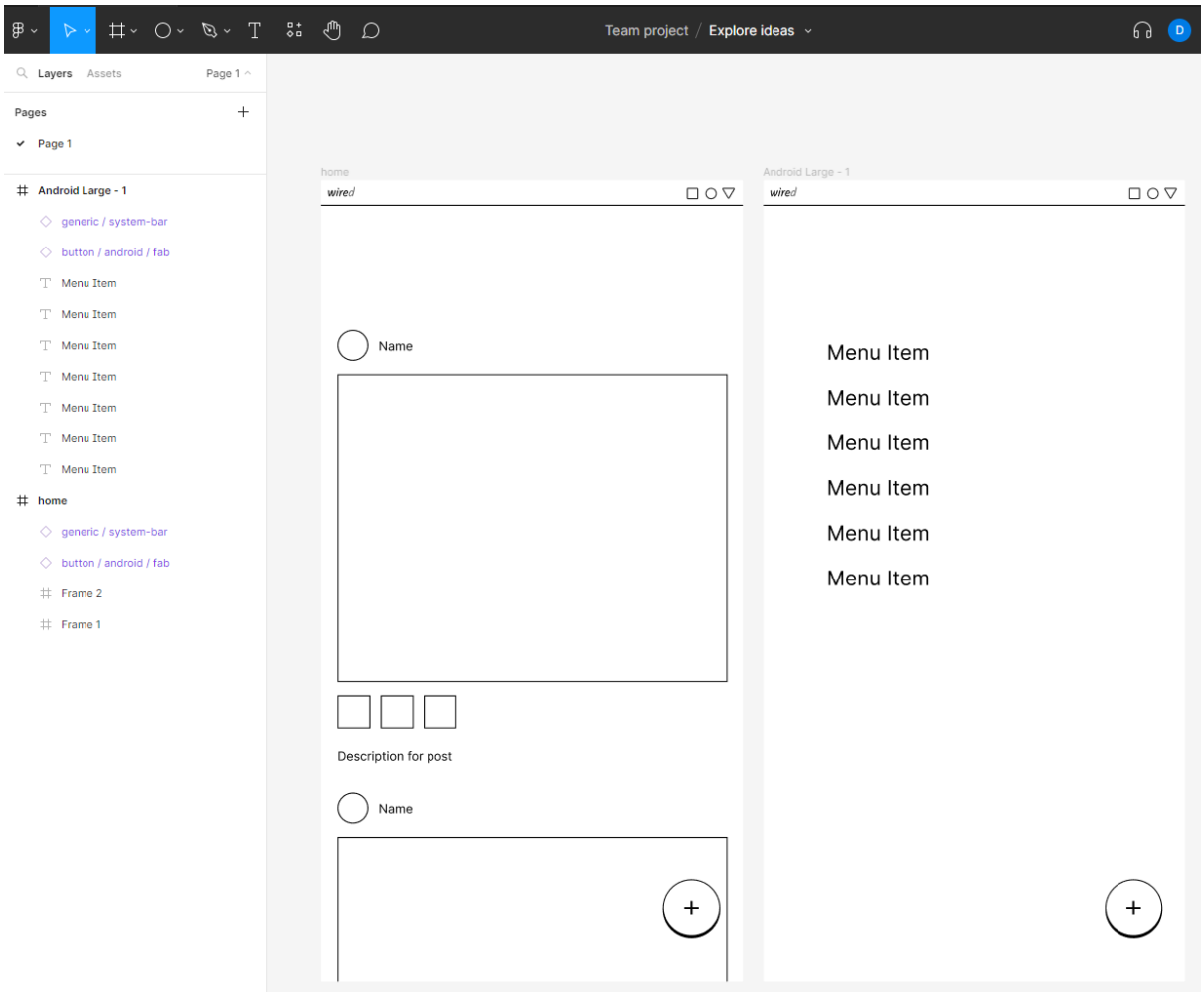
5. Additional screens

   Frame (f) → Google Pixel 2 XL → Text (t) → "Menu Item"

   → w 200 x h 23 글꼴크기 20

   복제 alt/option alt+Drag->cmd /ctrl D : 총 6 개 생성

   System-bar & fab → shift 키로 선택 → cmd/ctrl C → 프레임 선택 후 cmd/ctrl V



   텍스트 변경 : Home, Profile, Notifications, Settings, Search, Messaging

6. Iterations and feedback

프레임 복사하여 6 개 선택지 구성(menu1~menu6)

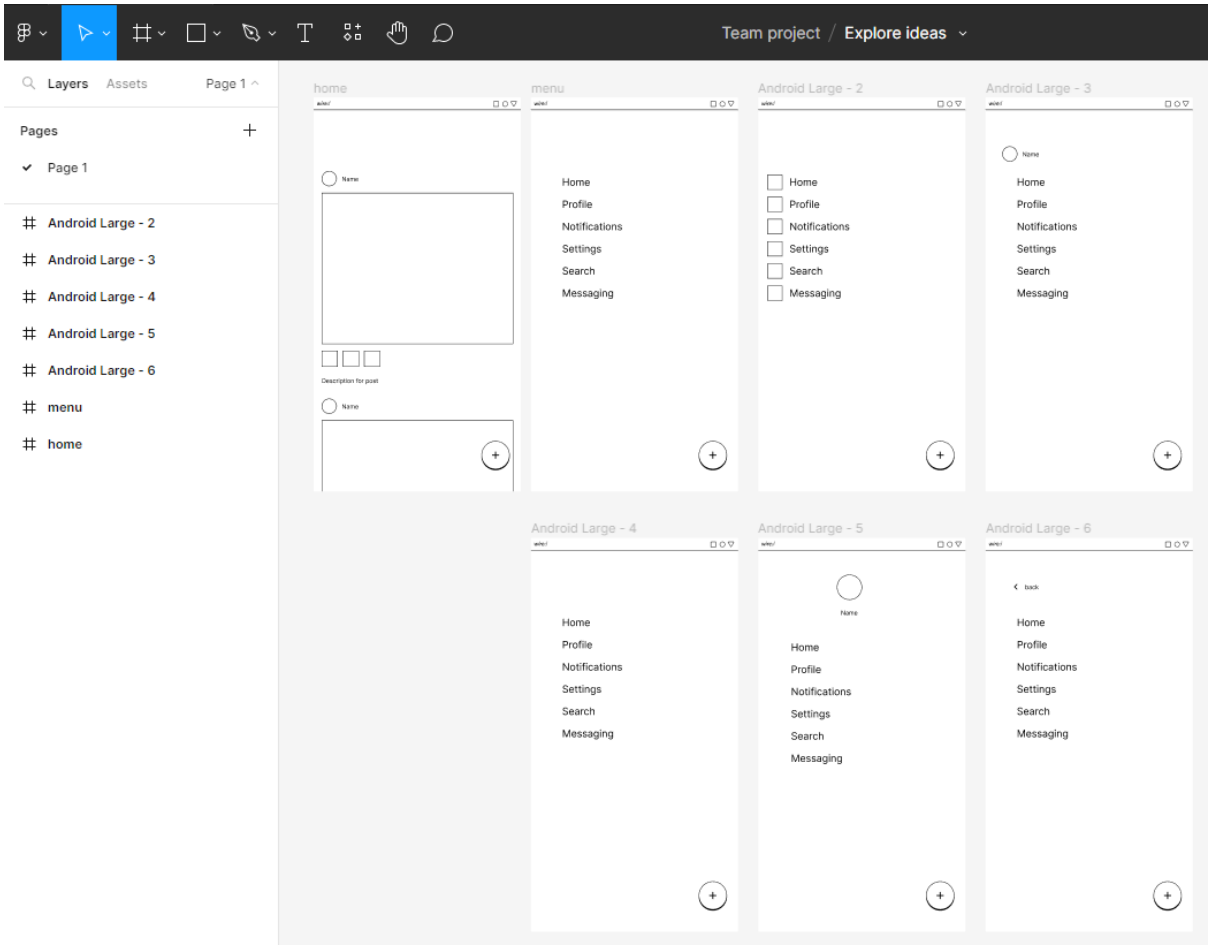
Menu2: Android / Controls / checkbox / android : Drag → Alt+Drag → ctrl+D

Menu3 : Checkbox copy-paste, Name 추가

Menu4 : 메뉴 이름 가운데 정렬(전체 선택 후 Text 가운데 정렬 사용)

Menu5 : Name 가운데로

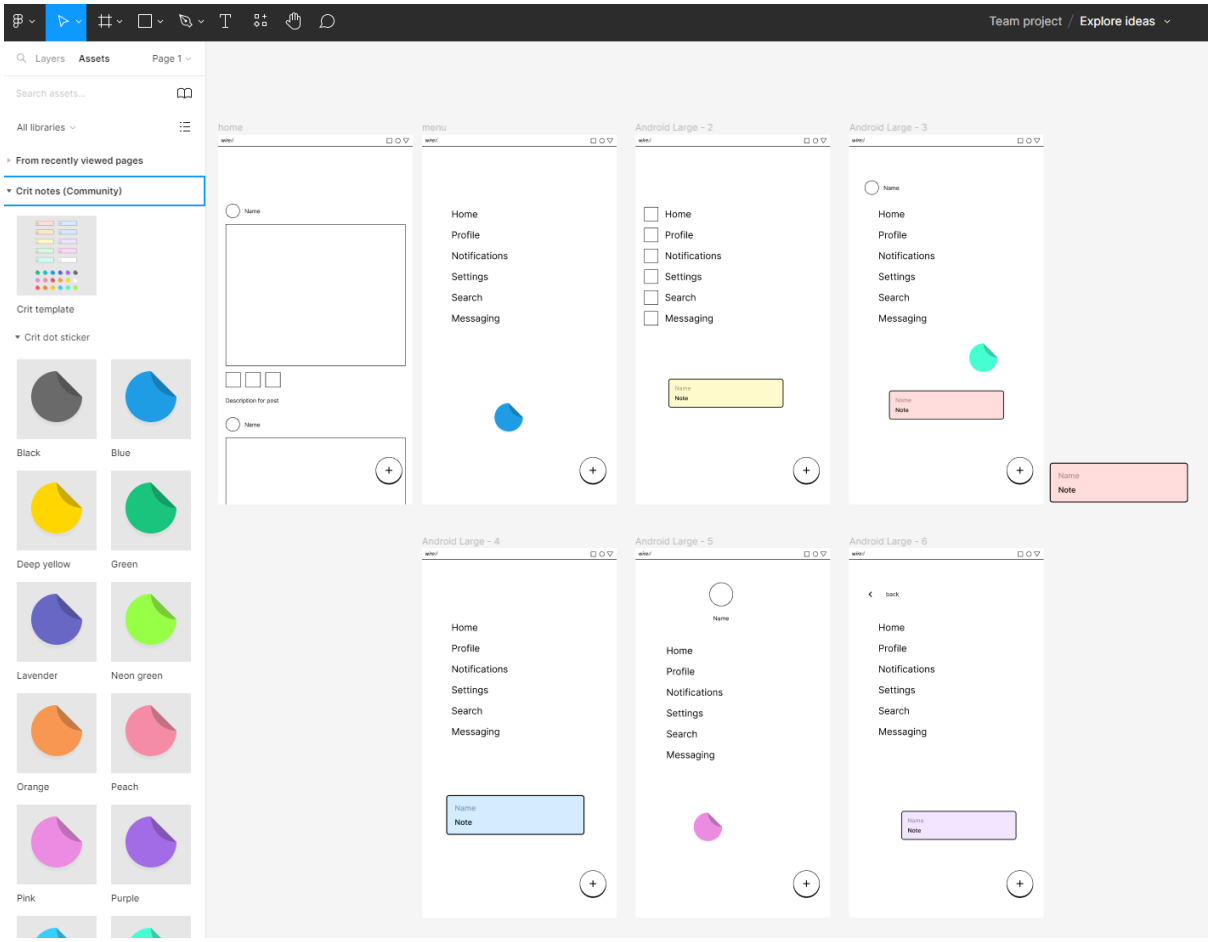
Menu6 : Android . Icons and symbols / Icon / android / 24 / back icon 추가



7. Share 버튼 - Copy link

asset crit notes

(community search → open in Figam → Move to Project → pulish → add to file)



2-2. Create designs

1. Pages + → Designs
2. Frame (f) → Google Pixel 2 XL (W 411, H 823)→ R w379, h 285 → Center 배치
3. alt/option 프레임 선택 - 마진 확인 → 32 (16)points margin

   → alt/option 프레임 선택 - 마진 확인

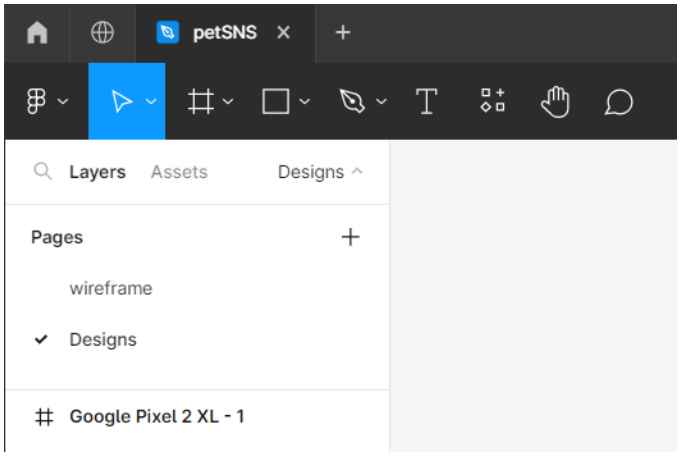
   Layout Grid 사용 : 8point - %보기 에서 켜고 끌수 있음(Shift G)
4. 사각형 선택 (프레임 내부의 하위 레이어 선택 시 Ctrl 키 사용) →코너 radius 8 로 변경
5. O - shift - 40x40 - left 정렬 - 상하 마진 8

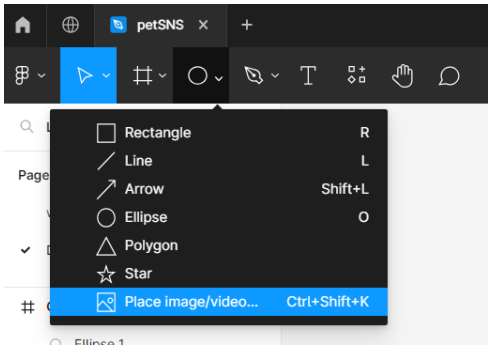
   (상하마진 정렬은 위쪽기준, alt 키 누리고 옵션 보면서 화살표로 이동)
6. Ctrl + Shift + K 이미지 추가 2 개 → 마우스로 추가 (이미지 다운 받아 놓을 것)

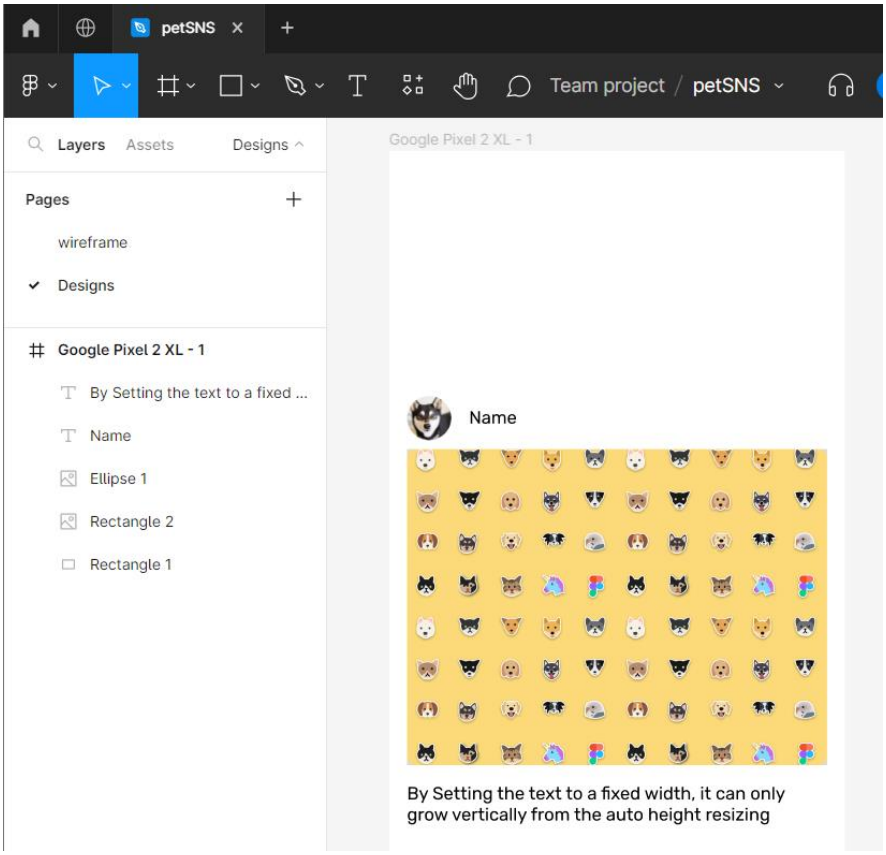
1-Avatar-placeholder.png

2-Post-placeholder.png

7. Text (t) → Name, Rubik, 16 →이미지와 중앙정렬, 마진 16
8. alt/option duplication → "By Setting the text to a fixed width, it can only grow vertically from the auto height resizing" → w 379 x h 38







10. 스타일 : 적용할 수 있는 재사용 속성 (많은 텍스트 요소를 한꺼번에 관리할 수 있다)

11. 스타일로 만들 텍스트 선택된 상태에서 → 텍스트 스타일 점 4 개 아이콘 클릭 → + 추가 → 스타일 이름 만들기(Body text) → 다른 텍스트에 스타일 적용

12. 텍스트 레이어 : 흰색프레임에 검은색 → 초고대비

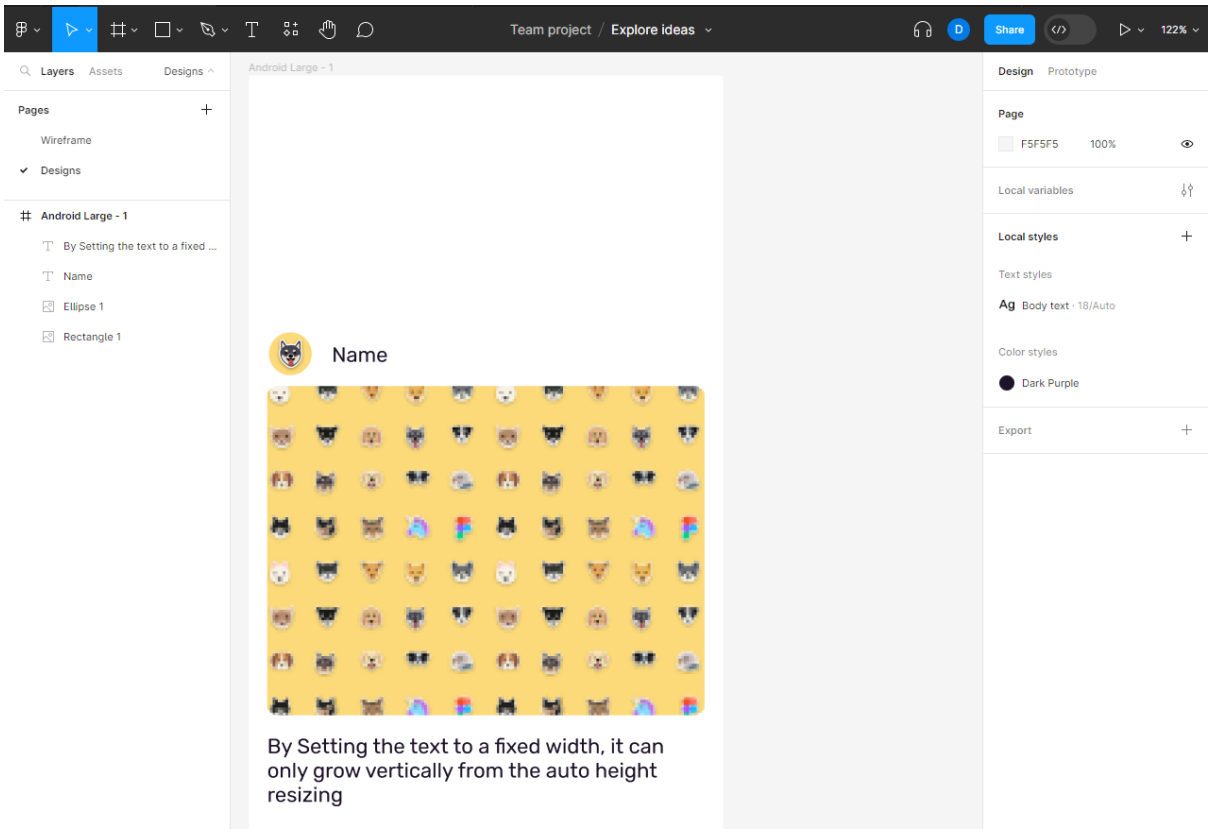
글씨 회색으로...ㅜㅜ (ABABAB)

텍스트 선택 후 마우스 오른쪽 메뉴 plugin → stark accessibility tools 플러그인(설치 및 로그인 필요) -> contrast checker → 1F132A 변경 → 컬러 스타일 설정 (Fill 점 4 개)

https://accessibility.naver.com/acc/guide_04

사이드바에서 모든 스타일 볼 수 있음

13. 텍스트 스타일 크기 18 조정 → 위치가 약간 어긋남, Auto Layout 필요한 이유

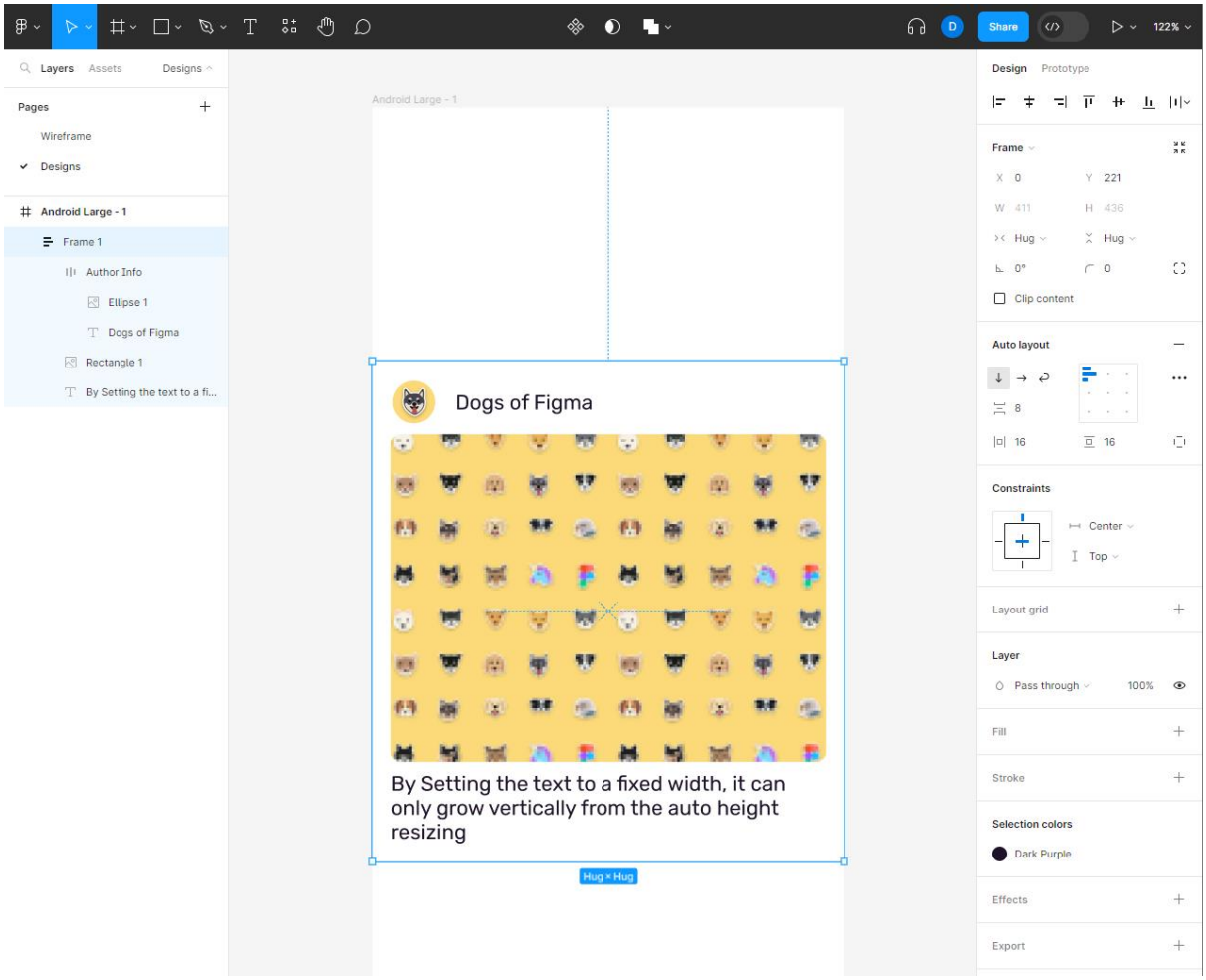


14. Auto Layout

모두 선택 → 가로 세로 레이아웃 변경 Shift A → "Name" 글씨도 세로로 정렬되는 경우 발생

→ Circle 과 "Name" 선택 후 → "Dogs of Figma" 글씨를 수평 Auto Layout 올 설정 후 전체 객체를 세로 Auto layout (shift A, 정렬 기준 조정)으로 설정한다.

전체 선택 후 Shift A → Auto Layout 마진 변경 8 → 텍스트 Resizing → fill + Hug (텍스트 길이에 맞게 텍스트 박스 변경) → 패딩 16 → 중앙 정렬



15. Create components (더 많은 같은 디자인 적용을 위해)

레이어 이름 변경(Post, Description, Image, Author, Name, avatar)

→ create component (option cmd K / Ctrl alt K) (Hug x Hug)

→ components page 추가 → move the components page

→ asset components 선택 (Local Components) → alt/opt Drag, Ctrl+D (모두 4 개)

→ clip contents 확장되는 레이어의 일부를 숨기고 싶을 때

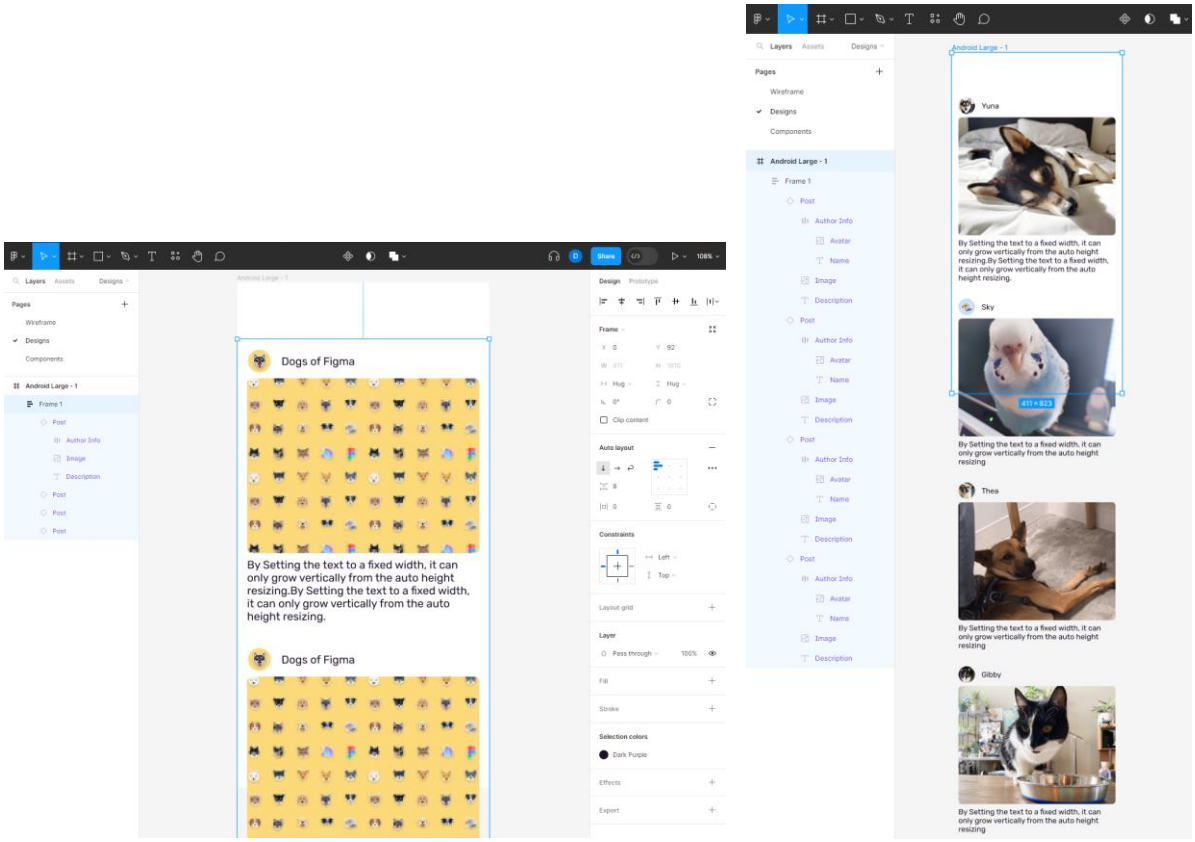
→ 4 개 components 모두선택 → Shift A (Auto Layout, Description 많아질 경우 대비)

→ 마진 8

→ 텍스트 길게 변경하면 auto layout 적용 → Frame rename "Feed"

→ 실제 컨텐츠(이미지) 추가 (cmd+shift+K/ctrl+shift+K) 인스턴스화(오버라이드)

: 1-Post, 1-Profile ~ 4-Post, 4-Profile



16. 버튼 → search Community - Petma Library 사용

17. 메뉴 아이콘

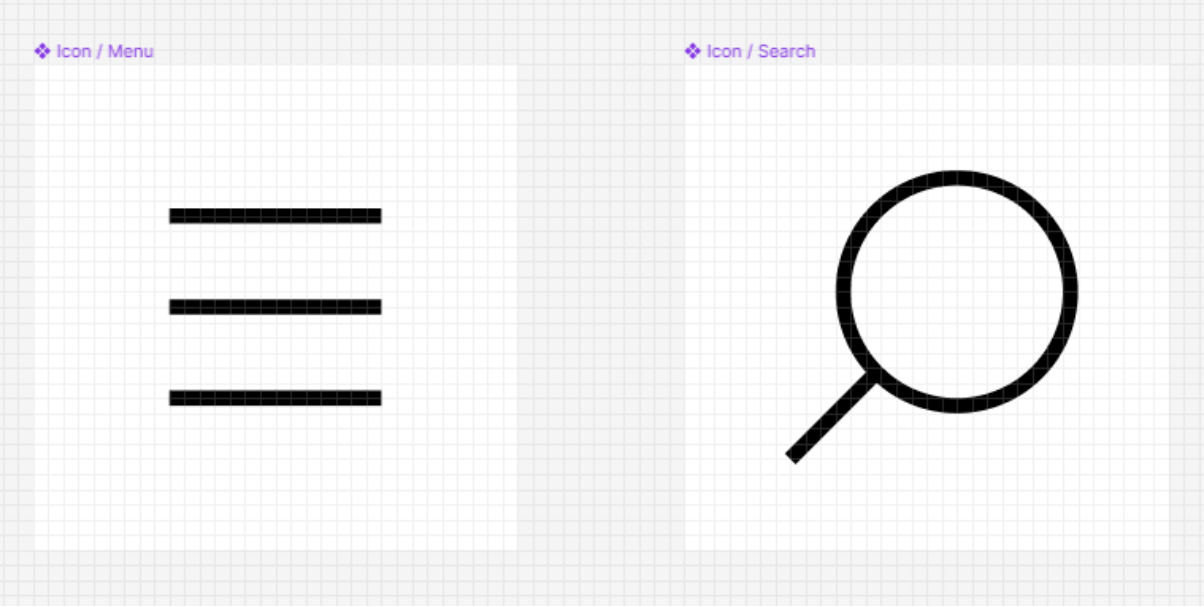
→ 32 x 32 프레임 생성 → shift 2 (zoom fit selection) → pen shift 수평선 enter (width 14)

→ 복제 3 개 , 간격 6 → 모두 선택 후 stroke - Round → 마우스로 중앙정렬

→ Create components → 이름변경 Icon/Menu → move the components page

18. 검색 아이콘

→ 32 x 32 프레임 생성 → 16 x 16 원 → shift x (swap fill and stroke) → shift+pen45(w 6)
→ 한쪽만 라운드 → 두 레이어 결합 Boolean operation → Union, Subtract, Intersect, Exclude
→ Create Component → 이름 변경 Icon/search → move to component page



19. 네비게이션 바

→ Component page 에서 411 x 80 프레임 생성

→ asset 에서 menu 아이콘을 Drag, 왼쪽+아래 마진 8

→ search 아이콘, constraint 조절(모서리)

→ 로고(Petma) → constraint center, 마진 8

→ status bar (in Petma) 상단 위치(위쪽정렬) → constraint left & right

→ 이름 변경 nav bar → create component(cmd option K, ctrl alt K)

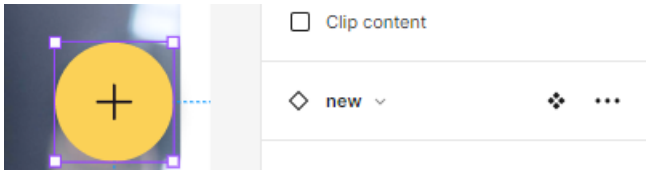
→ 디자인 페이지에 적용

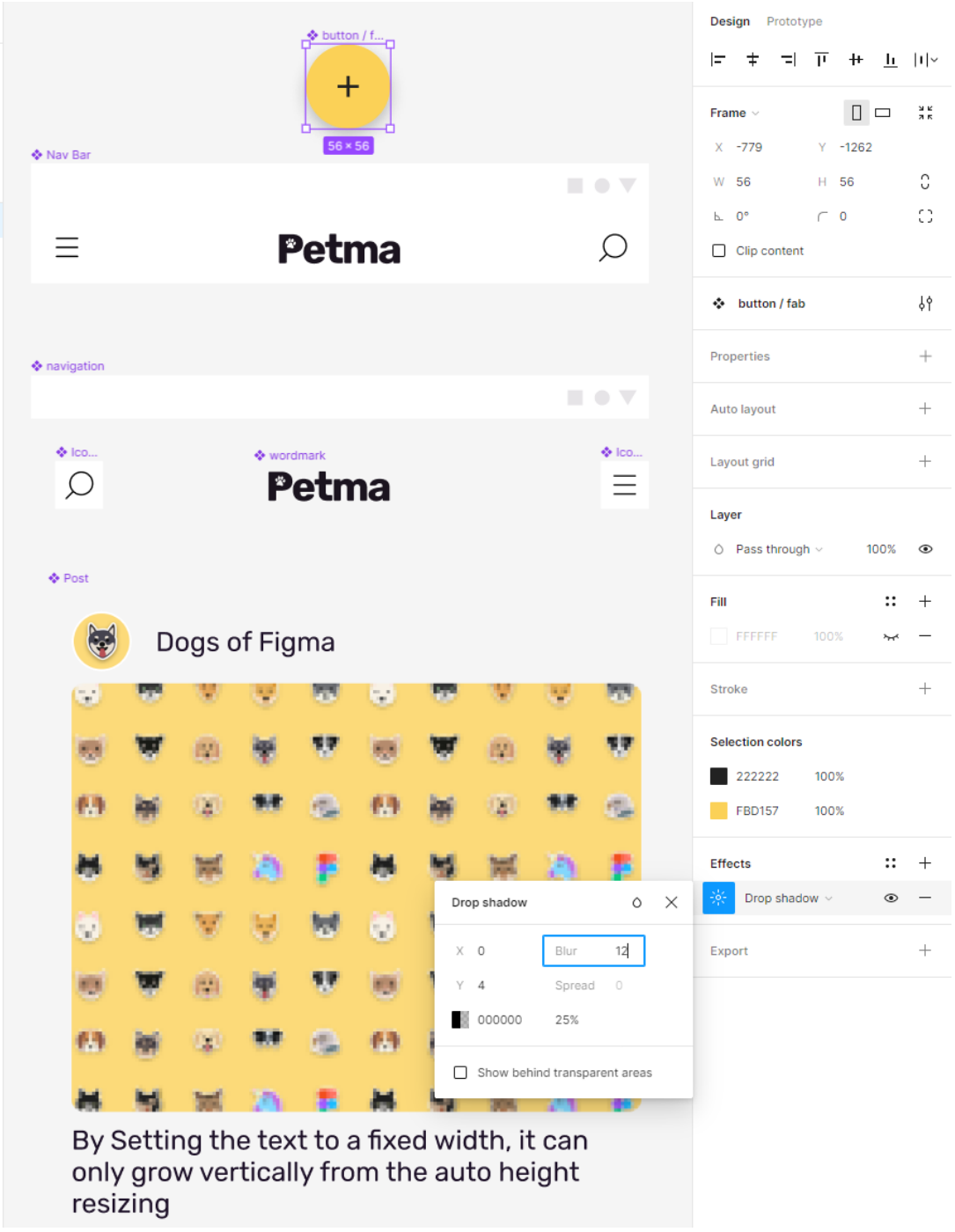
20. Floating action button

→ Petma library - Components / Buttons / fab / new 에서 Local Components 로 Drag

→ make component!

→ Component 의 fab 버튼 위치 시킴(프레임 따라서 위치 해야지 안 그러면 자동 레이아웃 때문에 수직 정렬 됨) → constraint 모서리 설정 → fab 선택 상태에서 사이드바 component 아이콘 선택 → 그림자 효과 (effect-drop shadow blur 12)



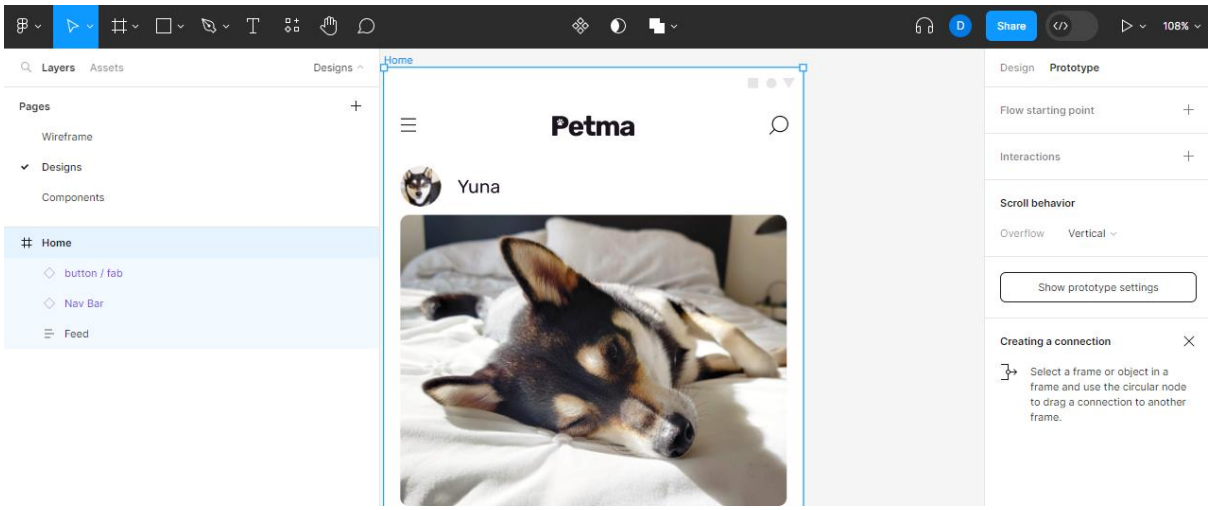


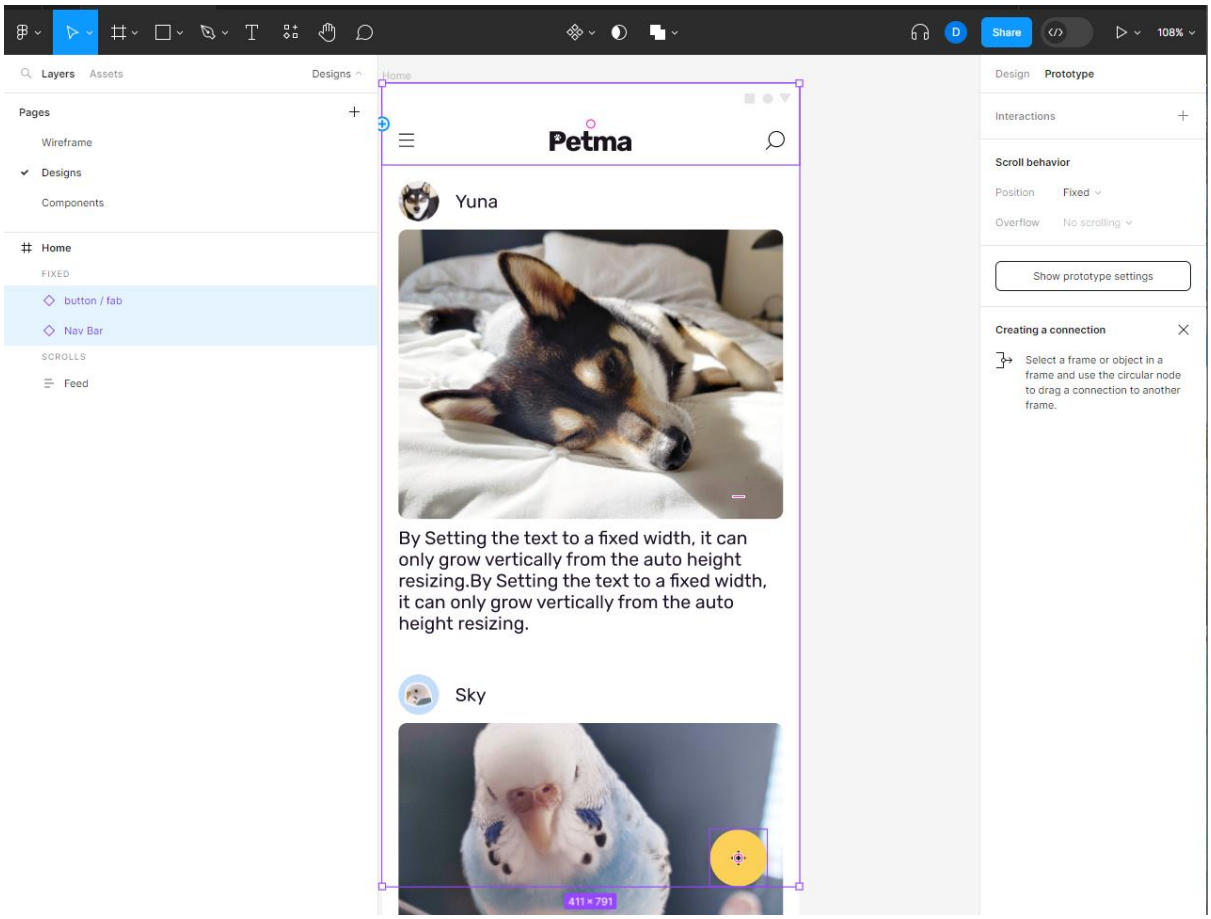
21. Scrolling and fix position

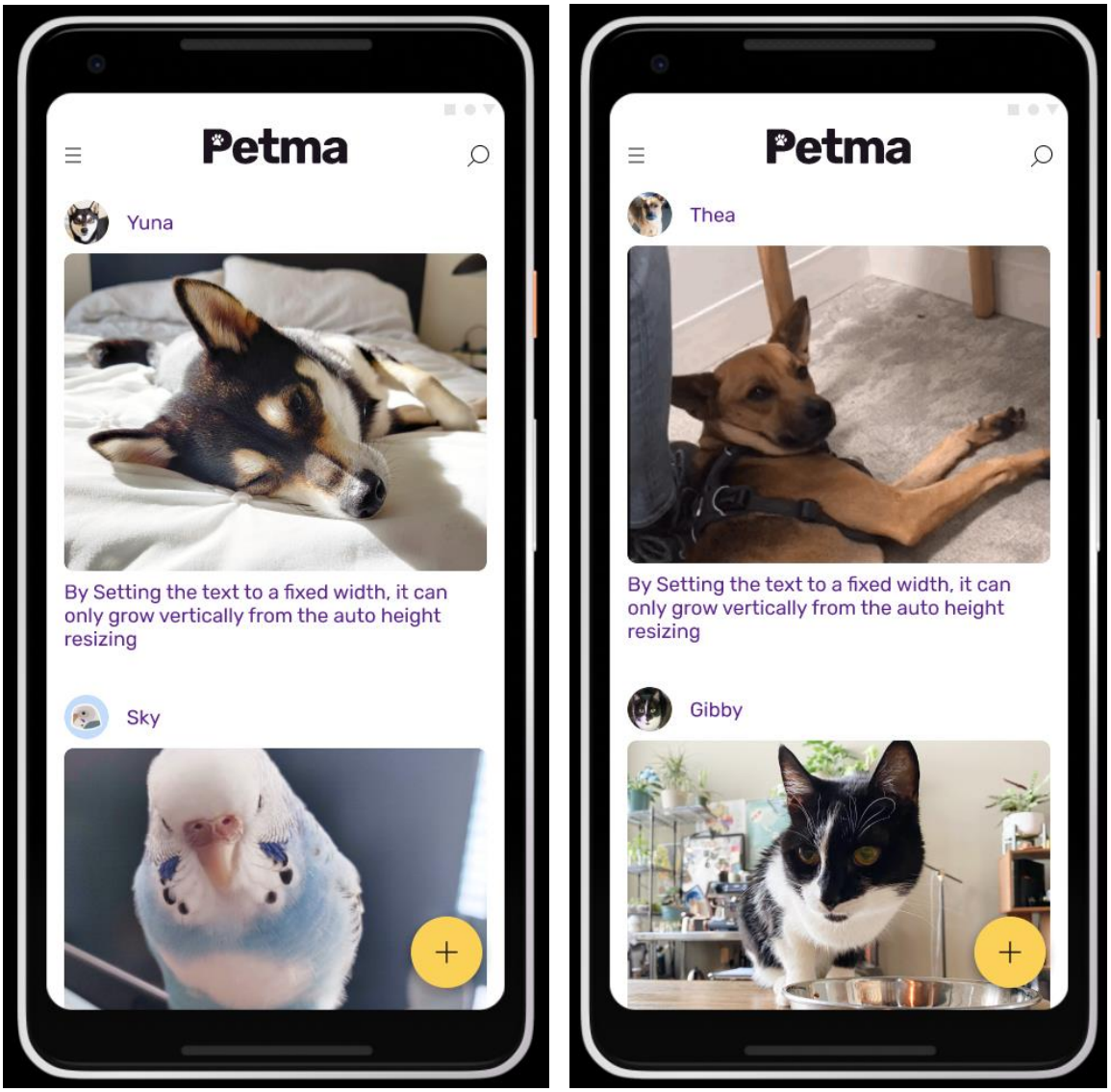
→ 프레젠테이션 뷰 → 재생 아이콘 (사이드바 Prototype, device : google pixel 2XL)

→ 프레임 선택 → Vertical Scroll

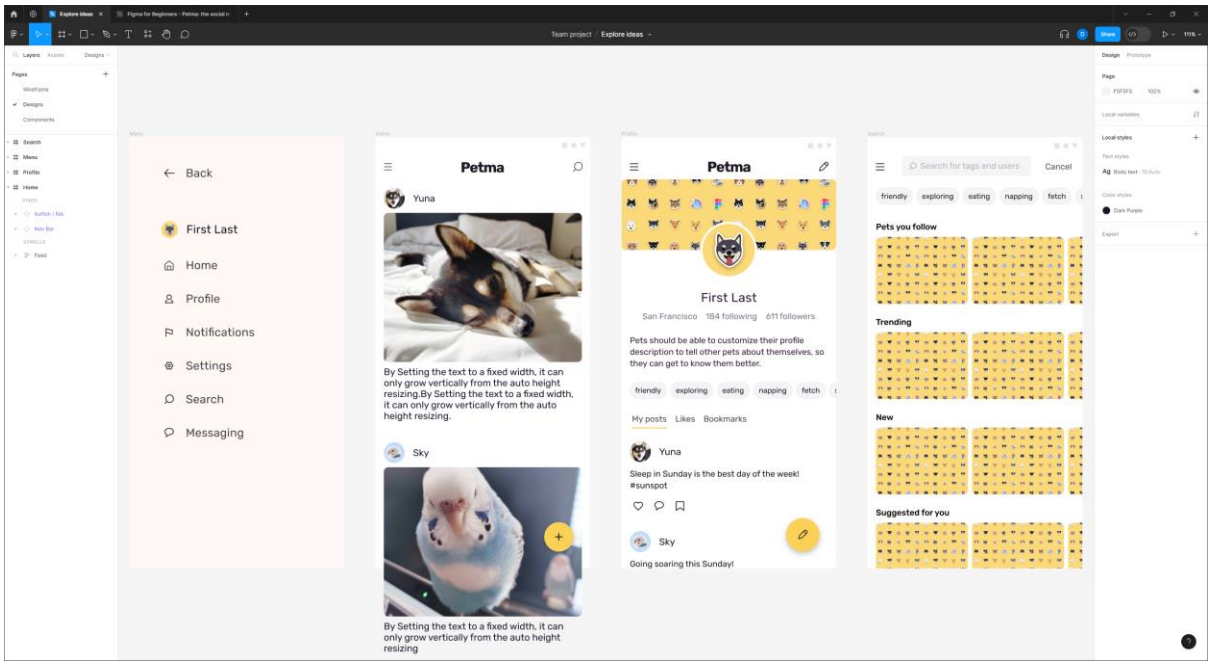
→ Status bar & Fab → scroll behavior : Fixed






Petma
Yuna
By Setting the text to a fixed width, it can only grow vertically from the auto height resizing
Sky
Petma
Thea
By Setting the text to a fixed width, it can only grow vertically from the auto height resizing
Gibby

2-3. Build prototypes

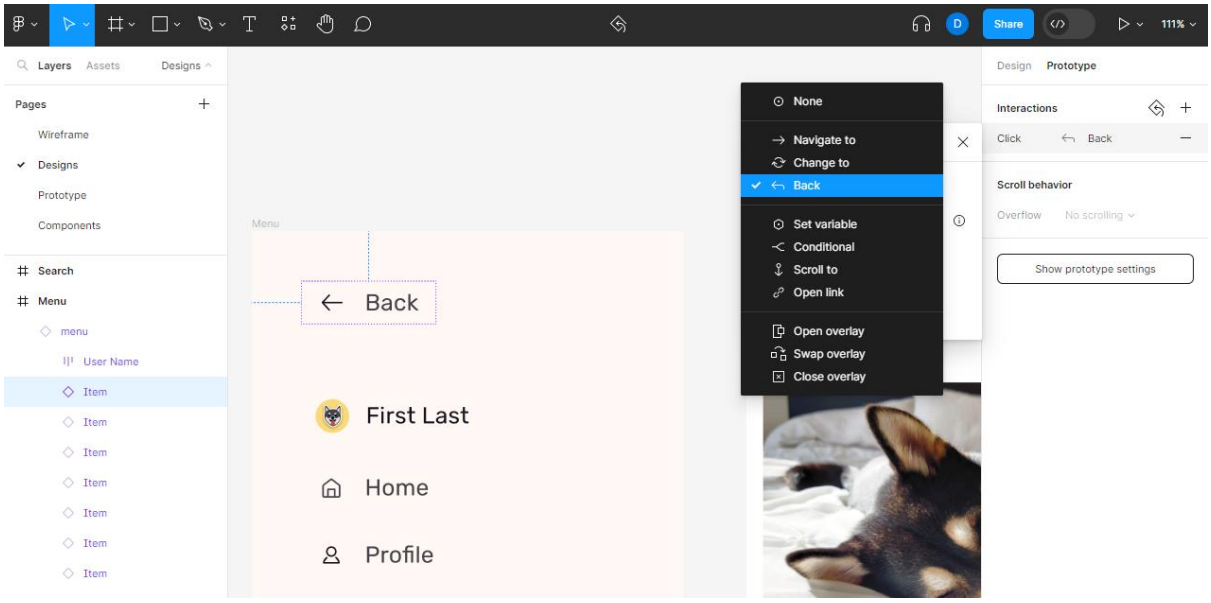


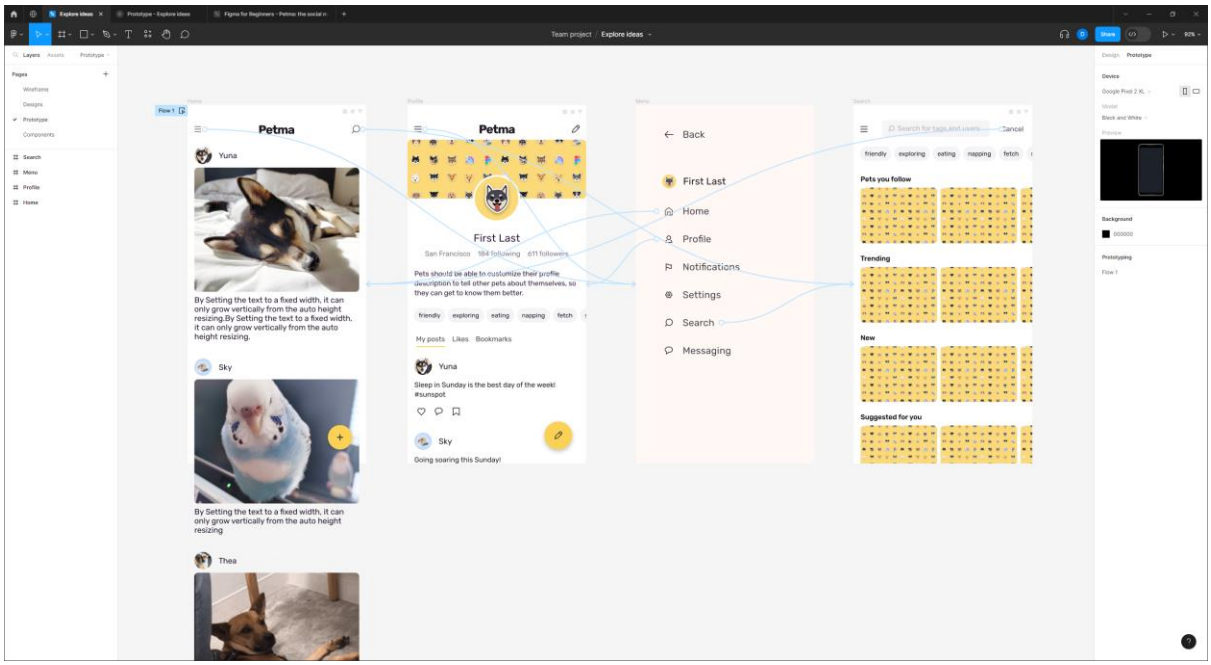
(Home 은 지금 만든 페이지, Profile, Menu, Search 페이지 복사 from Petma)

1. 페이지 duplicate : Designs → Prototype
2. 사이드바 : Prototype 모드 전환
3. Hotspot → interaction detail → destination
4. Menu 아이콘 더블클릭(핫스팟보임) → Menu 페이지 연결, Home 연결

   → Interaction (애니메이션 등 설정) : 기존에 존재하는 interaction 이 있으면 삭제하고 진행
5. Search 페이지의 메뉴 아이콘과 Profile 페이지의 메뉴아이콘을 메뉴 페이지에 연결함.
6. 메뉴페이지의 back 버튼 활성화 : 선택 → Interaction 추가
7. 프로토타입 피드백 → share prototype (anyone with the link, can view)

   → add comment (좌상단 말풍선) → submit → 편집기에서도 comment 볼 수 있음.
8. Petma library 에서 북마크 아이콘 찾아서 Drag 한 후 Create Component
9. 컴포넌트에 인스턴스 삽입 시 alt+Drag 로 할 것


Layers
Assets
Designs
Pages
Wireframe
Designs
Prototype
Components
Search
Menu
menu
User Name
Item
Item
Item
Item
Item
Item
Item
Menu
Back
First Last
Home
Profile
None
Navigate to
Change to
Back
Set variable
Conditional
Scroll to
Open link
Open overlay
Swap overlay
Close overlay
Design
Prototype
Interactions
Click
Back
Scroll behavior
Overflow
No scrolling
Show prototype settings


Petma
Yuna
By Setting the text to a fixed width, it can only grow vertically from the auto height resizing.By Setting the text to a fixed width, it can only grow vertically from the auto height resizing.
Sky
By Setting the text to a fixed width, it can only grow vertically from the auto height resizing
Thea
Petma
First Last
Pets should be able to customize their profile description to tell other pets about themselves, so they can get to know them better.
My posts
Likes
Bookmarks
Yuna
Sleep in Sunday is the best day of the week! #sunspot
Sky
Going soaring this Sunday!
Back
First Last
Home
Profile
Notifications
Settings
Search
Messaging
Pets you follow
Trending
New
Suggested for you

3. git

3-1. Version(Source) Control System

Terminology : 소스 코드의 변경 사항을 관리

버전관리(Version Control, Revision Control)

소스코드관리(Source Code Management, SCM)

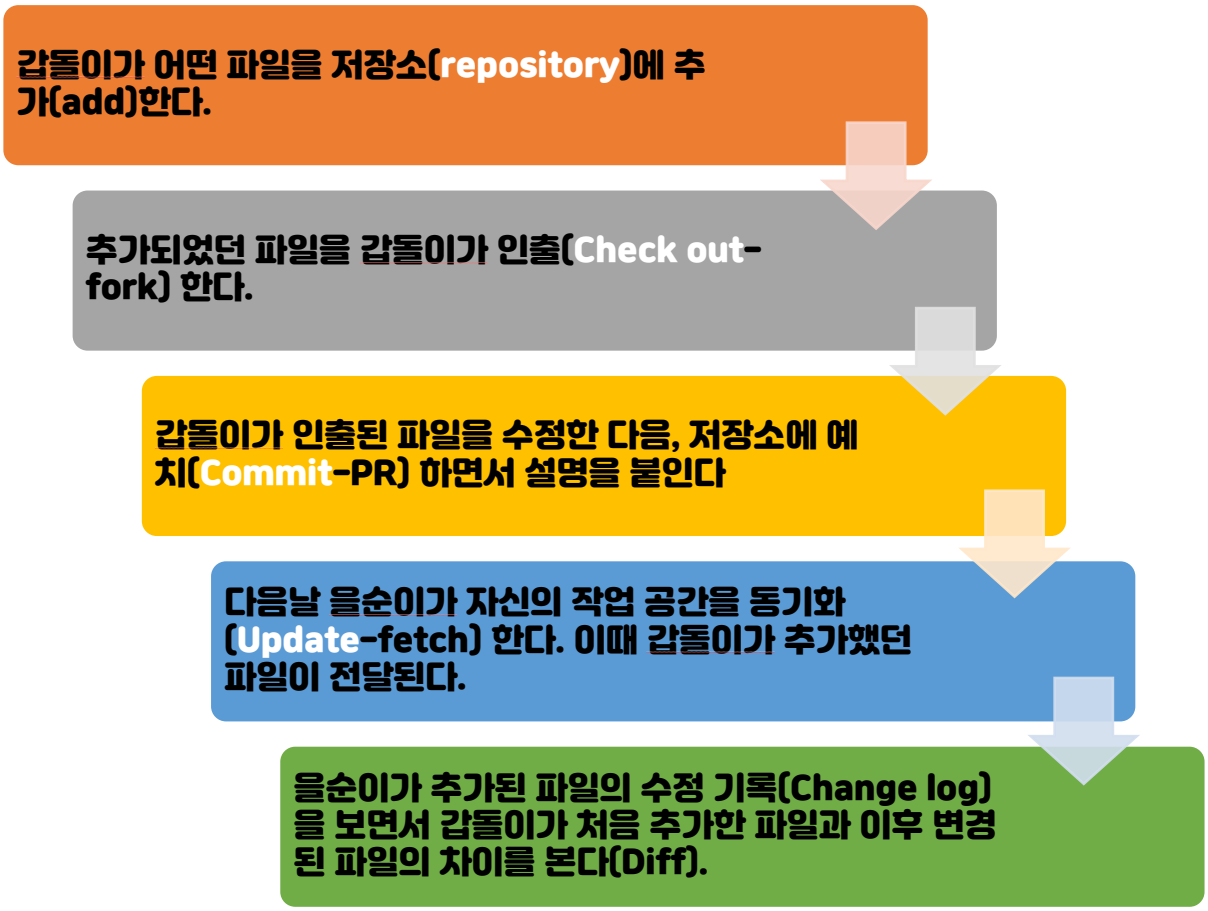
Software Version Management

형상관리(Software Configuration Management)

어떤 형태로든 소스 코드를 백업하여 분실의 위험에서 보호하고 개정 전후 내용을 파악하여 추후 발생할지도 모를 오류 수정에 대비하는 절차가 필요하다.
버전 관리 소프트웨어는 조직의 핵심 자산인 소스 코드의 개정과 백업 절차를 자동화하여 오류 수정 과정을 도와줄 수 있는 시스템이다.

https://ko.wikipedia.org/wiki/%EB%B2%84%EC%A0%84_%EA%B4%80%EB%A6%AC

3-2. Concept



3-3. SVN vs git

- SVN : 중앙집중식(CVCS)

  ✓ https://subversion.apache.org/features.html

  ✓ https://tortoisesvn.net/about.html

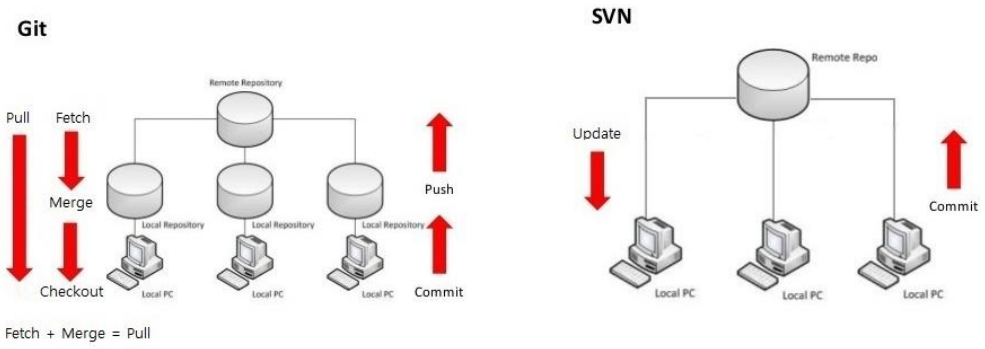
  ✓ https://ko.wikipedia.org/wiki/%EC%95%84%ED%8C%8C%EC%B9%98_%EC%84%9C%EB%B8%8C%EB%B2%84%EC%A0%84

- git : 분산관리식(DVCS)

  ✓ https://git-scm.com/

  ✓ https://ko.wikipedia.org/wiki/%EA%B9%83_(%EC%86%8C%ED%94%84%ED%8A%B8%EC%9B%A8%EC%96%B4)

Git over SVN



SVN	git
내 로컬 PC 에서 Commit 을 하면 **바로 중앙저장소에 반영** 내가 **Commit 을 하는 순간** 모든 사람에게 **공유가 된다.** 두 사람이 하나의 파일을 동시에 수정하고 커밋하였을 때 **충돌이 일어날 확률이 높다**	내 로컬 PC 에서 Commit 을 하면 로컬 저장소에 반영이 되고 **로컬저장소에서 Push 를 하면 원격저장소에 반영** 작업내용을 Commit 하여 로컬 저장소에 반영한 후, **원격저장소에서 fetch 로 로컬저장소로 마스터 파일을 받아와서 충돌이 나지 않게 merge** 를 한 다음, 로컬 저장소의 내용을 Push 하여 원격저장소에 올리면 그때 공유
	웹(Github) 상에 저장소 → 언제 어디서나 협업을 할 수 있고 중앙저장소에 에러가 생기면 모든 작업이 마비는 SVN 과 다르게 원격저장소에 에러가 생겨도 **로컬에서 복구하기 용이**하고, 히스토리 관리가 잘 제공되어 있어 **히스토리 관리가 용이**하다.
한번 개발하고 추가 개발이 적음	지속적, 반복적인 개발에 용이 (Agile Methodology)

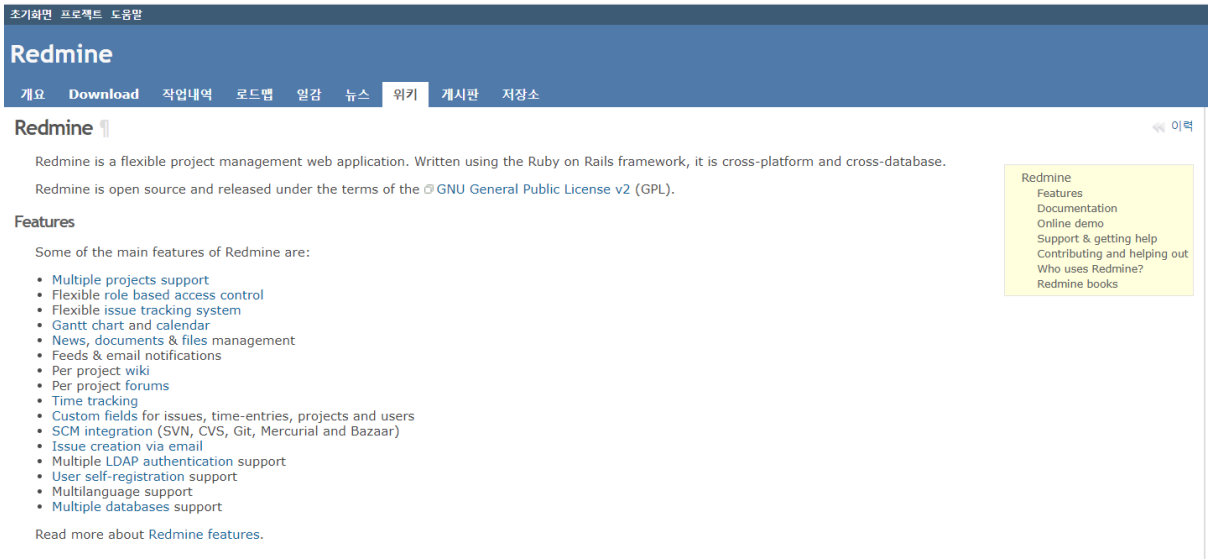
https://dzzienki.tistory.com/46

Git 장점	• **결정적 차이는 PR(pull request)에 있다.** • **Merge 를 위해 PR 을 함으로써 Code Review, Merge Conflict 해결에 있어 중앙집중식보다 낫다.** • **Version rollback 이 용이하다.**
Git 단점	• **중앙집중식보다 배우기 어렵다.** • **Version history 가 간결하지 않다.(rebase 등으로 관리 필요)**

APPENDIX : 형상관리 툴(Redmine)

: SCM 뿐만 아니라, 어플리케이션의 개발 계획 Gantt chart, Issue 관리, Documentation 기능까지 통합

- https://www.redmine.org/



APPENDIX : Linus Torvalds

https://en.wikipedia.org/wiki/Linus_Torvalds

Linus Benedict Torvalds (/ ˈ l iː n ə s ˈ t ɔːr v ɔː l d z / *LEE -nəs TOR -vawldz* , [3] 핀란드 스웨덴어: ['liːnʉs ˈtuːrvɑlds] 나는 ; 1969년 12월 28일 출생)은 Debian , Arch 및 Android 와 같은 Linux 배포판 에서 사용되는 Linux 커널 의 창시자이자 수석 개발자인 핀란드의 소프트웨어 엔지니어입니다. 그는 또한 분산 버전 제어 시스템 Git을 만들었습니다 .

APPENDIX : Github

https://ko.wikipedia.org/wiki/%EA%B9%83%ED%97%88%EB%B8%8C

깃허브

문서 토론

위키백과, 우리 모두의 백과사전.

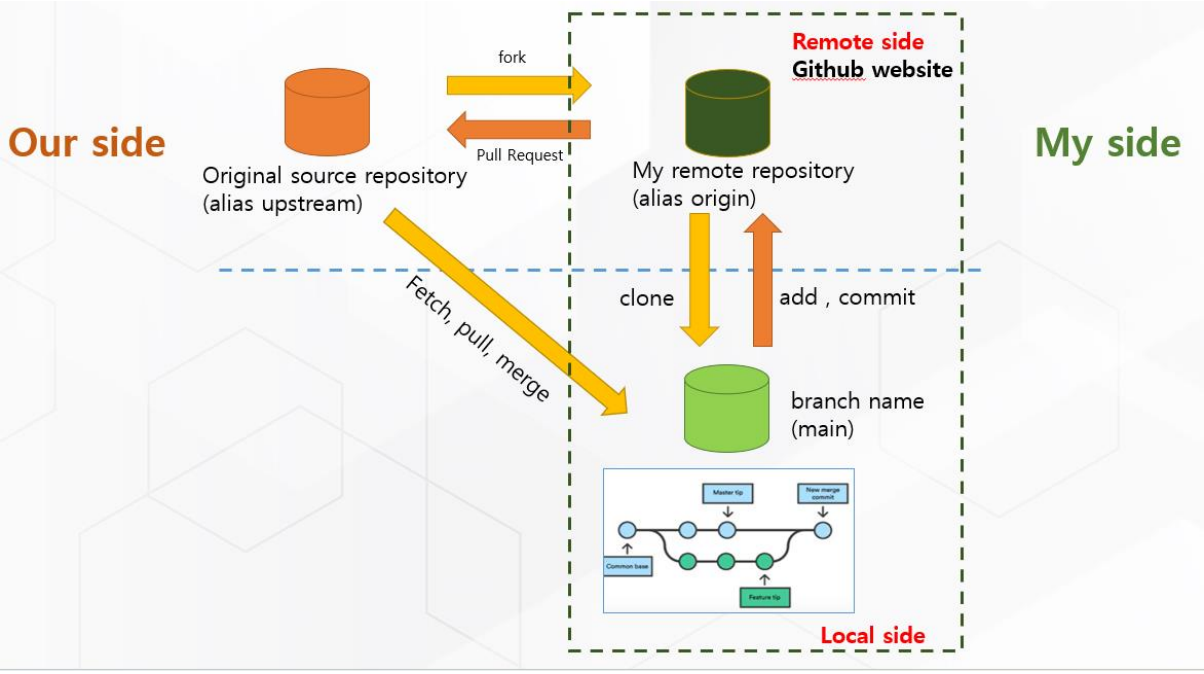
깃허브(GitHub, /ˈgɪtˌhʌb/, 원래 이름: Logical Awesome LLC)[1]는 루비 온 레일스로 작성된 분산 버전 관리 툴인 깃 저장소 호스팅을 지원하는 웹 서비스이다. 깃허브는 영리적인 서비스와 오픈소스를 위한 무상 서비스를 모두 제공한다. 2009년의 깃 사용자 조사에 따르면 깃허브는 가장 인기있는 깃 저장소 호스팅 서비스이다.[2] 또한 2011년의 조사에서도 가장 인기있는 오픈 소스 소프트웨어 인터넷 호스팅 서비스로 꼽혔다.[3]

깃이 텍스트 명령어 입력 방식인데 반해, 깃허브는 그래픽 유저 인터페이스(GUI)를 제공한다. 깃허브는 페이스트빈(pastebin)과 유사한 서비스인 기스트(Gist)와 위키를 각 저장소마다 운영하고 있으며, 깃 저장소를 통해 고칠 수 있다.

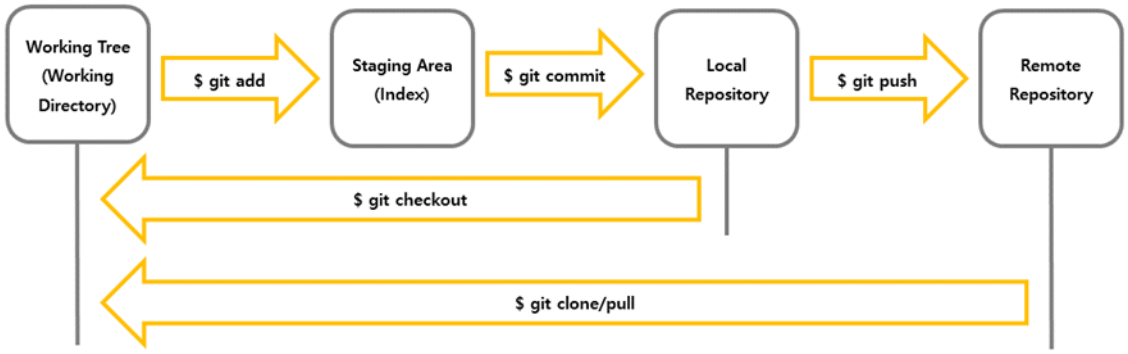
깃허브 회사는 2008년 톰 프레스턴워너(Tom Preston-Werner), 크리스 완스트래스(Chris Wanstrath), 피제이 하이엣(PJ Hyett)이 공동 설립했다. (Andreessen Horowitz) 등에서 투자를 받았다. 2010년 1월부터 깃허브는 GitHub, Inc. 라는 이름으로 운영되고 있다.[4] 깃허브의 마스코트는 고양이 머리에 문어 다리가 달린 옥토캣(Octocat)이다. 본사는 미국 캘리포니아주 샌프란시스코에 있다.

2018년 6월 4일, 마이크로소프트는 75억 달러에 깃허브를 인수하였다[5]

Git Concept



Git Structure



1. working tree (working directory)

working tree 혹은 working directory 는 말 그대로 현재 작업 공간을 뜻한다. 이러한 working tree 는 프로젝트의 특정 버전을 checkout 한 것이다.

2. staging area (index)

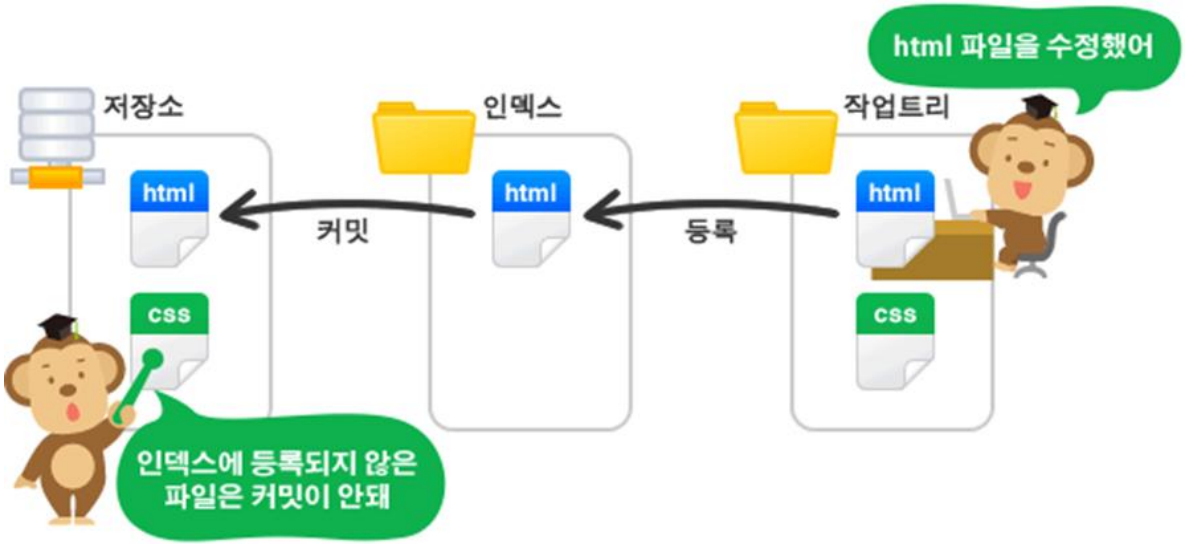
staging area 는 index 라고도 불리며, Git directory 에 포함된 파일이다. 다음에 commit 할 파일들이 저장되어 있는 공간으로 이때, 파일들은 $ git add 명령어를 통해 staging area 에 올라간다.

3. git directory (repository)

git repository 에는 local repository 와 remote repository 가 존재한다. Local repository 는 개인 PC 에 파일을 저장하는 저장소이며, Remote Repository 는 원격저장소로 공유를 위한 저장소로 동료들과의 협업을 가능하게 해준다.

remote repository 를 $ git clone 을 통해 받아오면 내 PC 에 local repository 가 만들어지게 된다.

이에 각각의 개발자는 우선 local repository 에 작업 내용을 저장하며 개발을 진행하게 된다. 그리고 $ git push 명령어를 통해 작업내용을 동료들과 공유할 수 있다.



https://backlog.com/git-tutorial/kr/

git command

- https://education.github.com/git-cheat-sheet-education.pdf
- https://git-scm.com/docs

git & GitHub Install guide

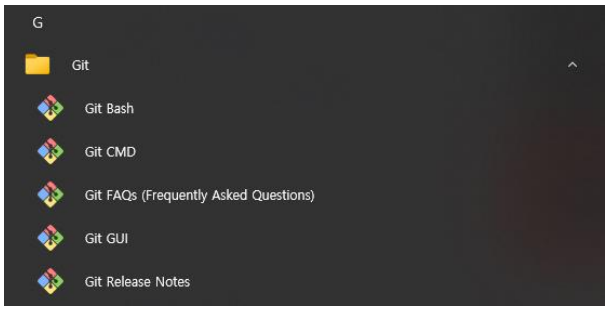
- https://whalec.io/git/git-git-%EC%84%A4%EC%B9%98%ED%95%98%EA%B8%B0-windows/
- https://www.lainyzine.com/ko/article/how-to-install-latest-version-of-git-on-macos/

windows

Mac

git 설치 후 확인

```
$ git --version
git version 2.39.0.windows.2
```



git update

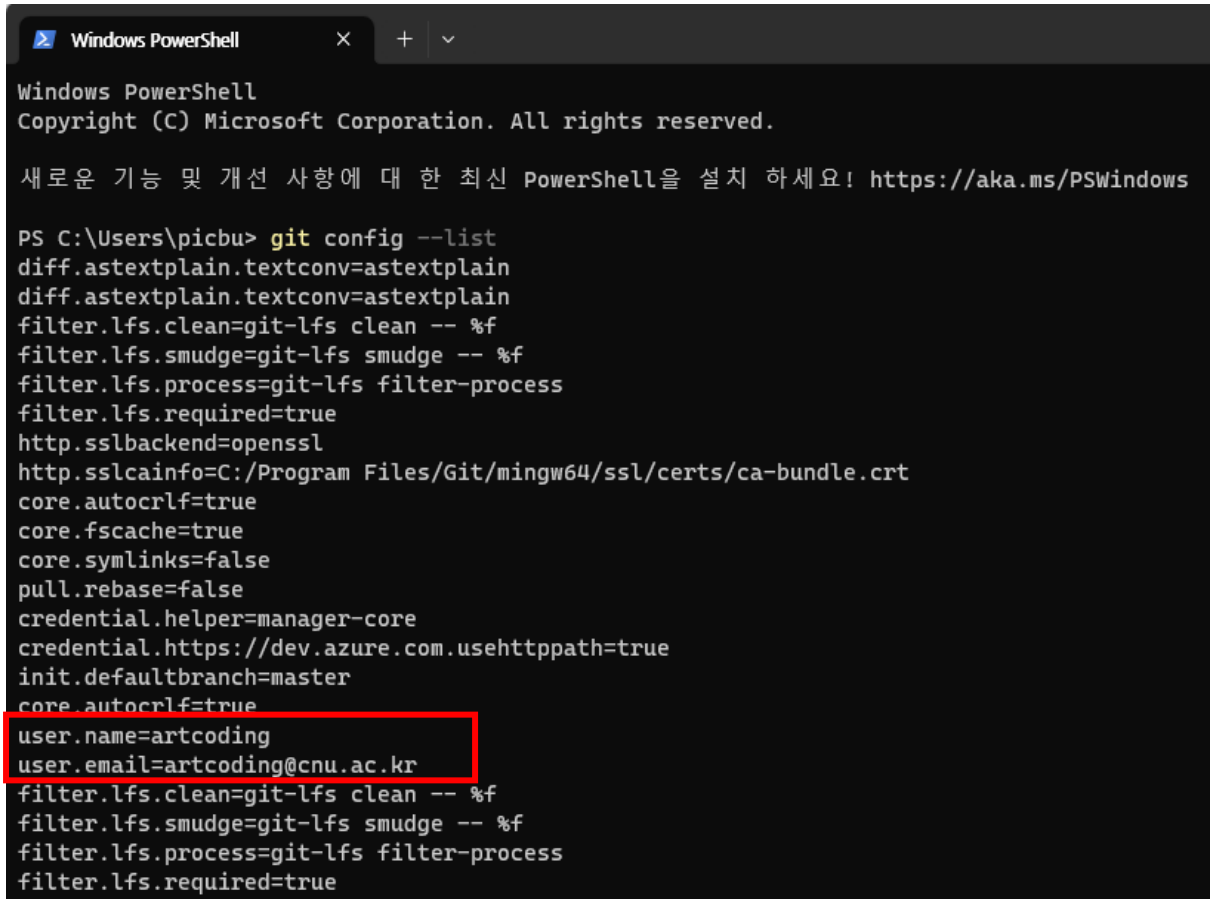
```
$ git update-git-for-windows
```

```
PS C:\Users\picbu> git --version
git version 2.35.1.windows.2
PS C:\Users\picbu> git update-git-for-windows
Git for Windows 2.35.1.windows.2 (64-bit)
Update 2.42.0.windows.2 is available
Download and install Git for Windows v2.42.0.windows.2 [N/y]? y
########################################################################
######################################## 100.0%
PS C:\Users\picbu> git --version
git version 2.42.0.windows.2
PS C:\Users\picbu>
```

TIPS : Micorsoft store 에서 “powershell” 검색하여 유틸 설치하면 편함

config setting

1. Github signup
2. Git 설치하기 : https://git-scm.com/downloads
3. 설치 완료 후 Git bash 또는 터미널 열기
4. git 환경설정 하기
    - Step 1 : 유저이름 설정
        - git config --global user.name "your_name"
    - Step 2 : 유저 이메일 설정하기
        - git config --global user.email "your_email"
        - Github 가입시 사용한 이메일을 써주세요!
    - 잘못 기입 시 삭제하기
        - git config --unset --global user.name
    - Step 3 : 정보 확인하기
        - git config --list



// global 로 default 브랜치를 main 으로 설정하기

```
$ git config --global init.defaultBranch main
```

// defaultBranch 값 읽기

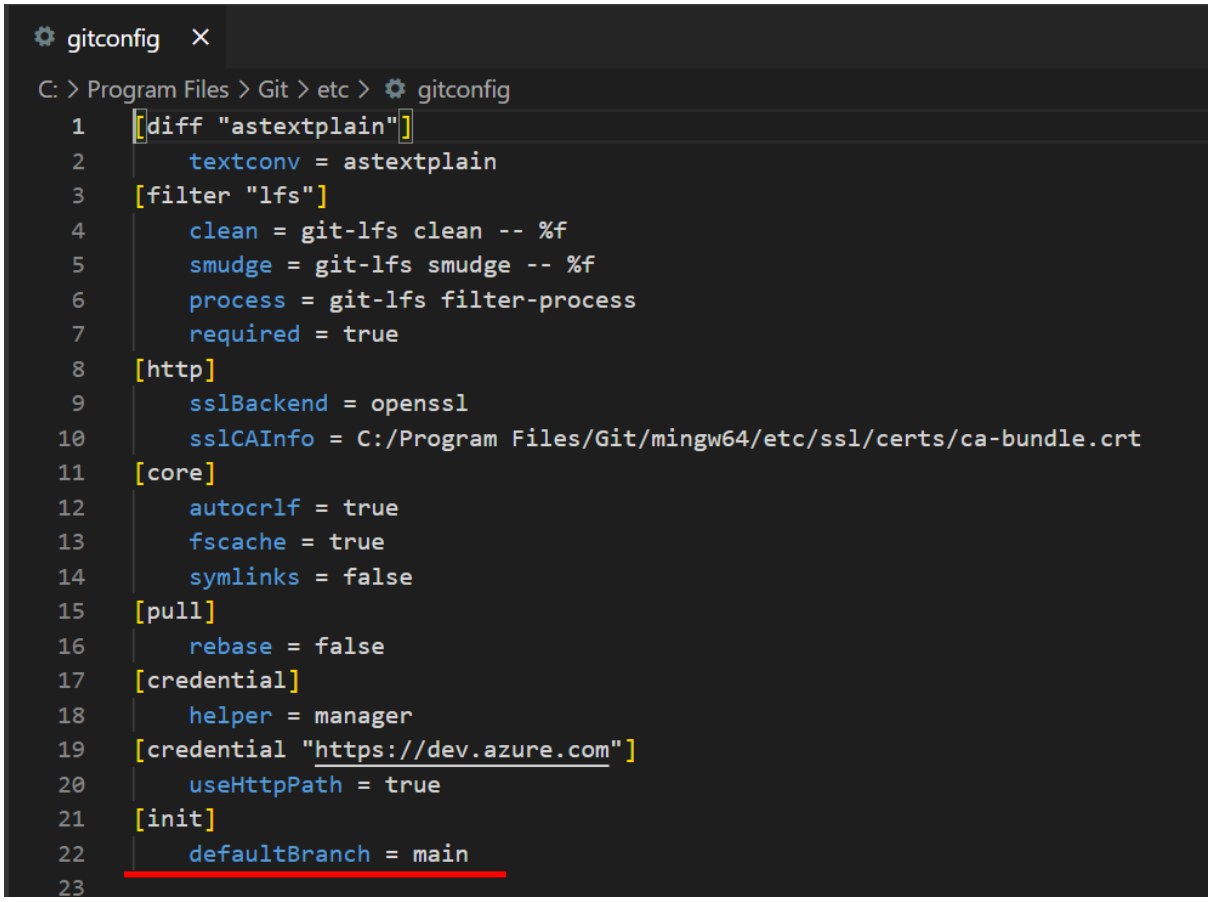
```
$ git config --get  init.defaultBranch

> main
```

// 출처 확인

```
git config -l --show-origin
```

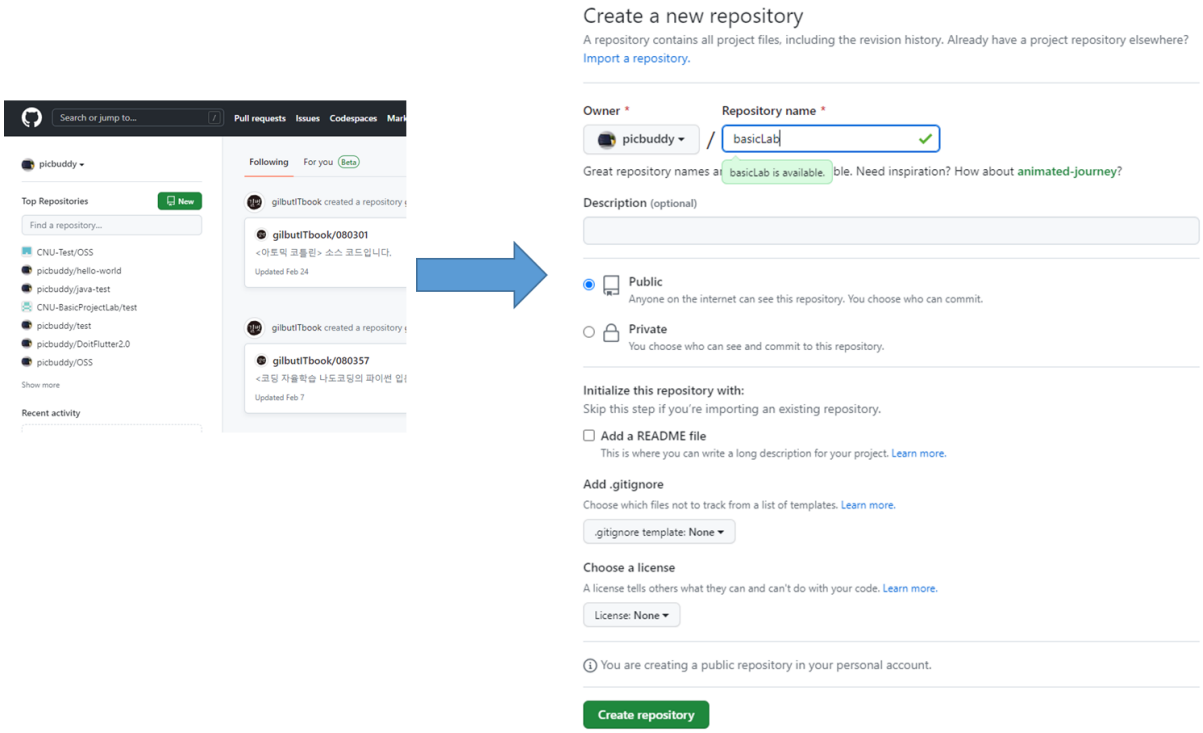
```
PS C:\Users\picbu> git config -l --show-origin
file:C:/Program Files/Git/etc/gitconfig diff.astextplain.textconv=astextplain
file:C:/Program Files/Git/etc/gitconfig filter.lfs.clean=git-lfs clean -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Program Files/Git/etc/gitconfig filter.lfs.process=git-lfs filter-process
file:C:/Program Files/Git/etc/gitconfig filter.lfs.required=true
file:C:/Program Files/Git/etc/gitconfig http.sslbackend=openssl
file:C:/Program Files/Git/etc/gitconfig http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
file:C:/Program Files/Git/etc/gitconfig core.autocrlf=true
file:C:/Program Files/Git/etc/gitconfig core.fscache=true
file:C:/Program Files/Git/etc/gitconfig core.symlinks=false
file:C:/Program Files/Git/etc/gitconfig pull.rebase=false
file:C:/Program Files/Git/etc/gitconfig credential.helper=manager
file:C:/Program Files/Git/etc/gitconfig credential.https://dev.azure.com.usehttppath=true
file:C:/Program Files/Git/etc/gitconfig init.defaultbranch=master
file:C:/Users/picbu/.gitconfig  core.autocrlf=true
file:C:/Users/picbu/.gitconfig  user.name=picbuddy
file:C:/Users/picbu/.gitconfig  user.email=picbuddy@naver.com
file:C:/Users/picbu/.gitconfig  filter.lfs.clean=git-lfs clean -- %f
file:C:/Users/picbu/.gitconfig  filter.lfs.smudge=git-lfs smudge -- %f
file:C:/Users/picbu/.gitconfig  filter.lfs.process=git-lfs filter-process
file:C:/Users/picbu/.gitconfig  filter.lfs.required=true
file:C:/Users/picbu/.gitconfig  credential.helperselector.selected=manager-core
file:C:/Users/picbu/.gitconfig  difftool.sourcetree.cmd='' "$LOCAL" "$REMOTE"
file:C:/Users/picbu/.gitconfig  mergetool.sourcetree.cmd=''
file:C:/Users/picbu/.gitconfig  mergetool.sourcetree.trustexitcode=true
file:C:/Users/picbu/.gitconfig  init.defaultbranch=main
file:.git/config        core.repositoryformatversion=0
file:.git/config        core.filemode=false
file:.git/config        core.bare=false
file:.git/config        core.logallrefupdates=true
file:.git/config        core.symlinks=false
file:.git/config        core.ignorecase=true
file:.git/config        remote.helloworld.url=https://github.com/CNU-Test/hello-world.git
file:.git/config        remote.helloworld.fetch=+refs/heads/*:refs/remotes/helloworld/*
file:.git/config        remote.upstream.url=https://github.com/picbuddy/hello-world.git
file:.git/config        remote.upstream.fetch=+refs/heads/*:refs/remotes/upstream/*
```

Gitconfig 파일을 관리자 권한으로 수정하기

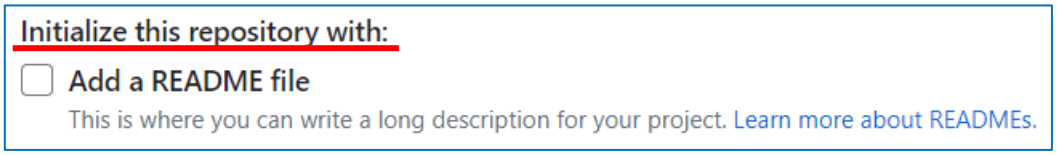


3-4. Work alone with Github

Step #1. Create a New Repository on GitHub



- 체크 안 하면 Github 에서는 Repo 빈방을 만드는 것이고 Local 에서 git 을 initialize 하여 remote 로 commit 하는 과정으로 진행 된다.
- 체크하게 되면 Github 에서 git 이 initialize 되기 때문에 Local 에서는 clone 받아서 진행하는 것이 좋다.
- Github 에서 initialize 된 git 과 Local 에서 initialize 된 git 을 강제로 병합하는 방법도 있다.



체크한 경우 Local 과 Remote 의 Git history 가 달라진다.

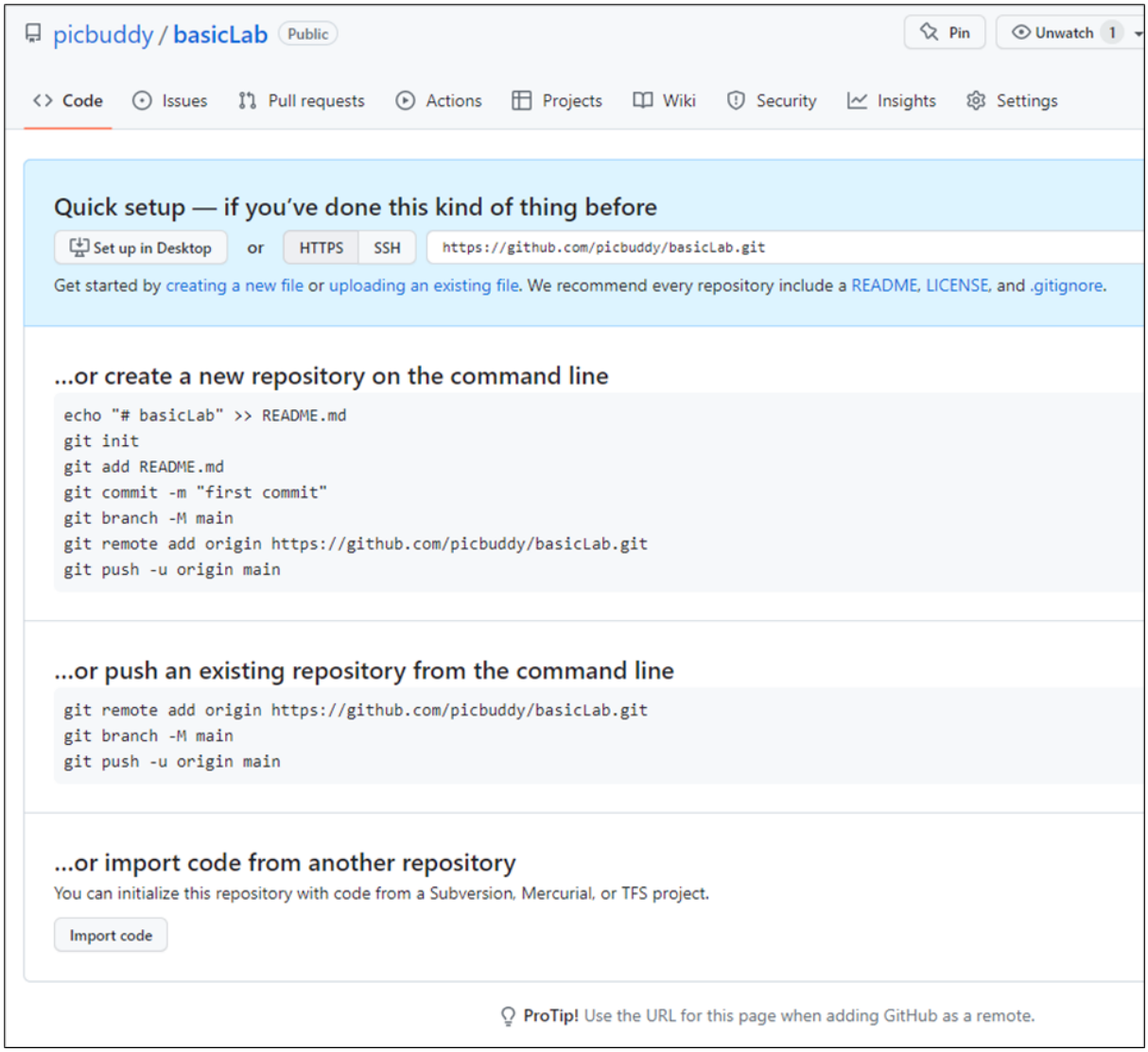
체크한 경우 push 하기 위해서는

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

다른 history 를 가진 프로젝트를 허용하는 옵션으로 Local 과 Remote 를 동기화 후

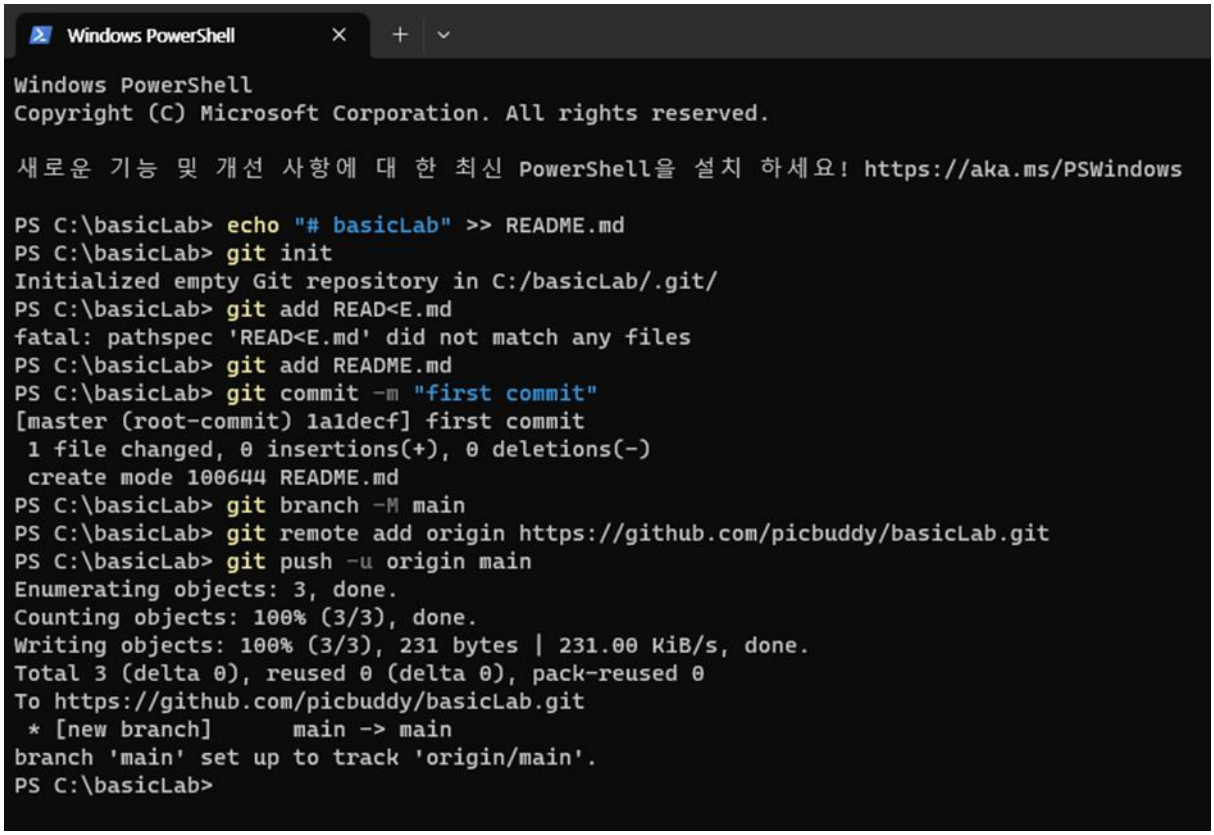
```
$ git push origin main
```

Step #2. create first file & commit



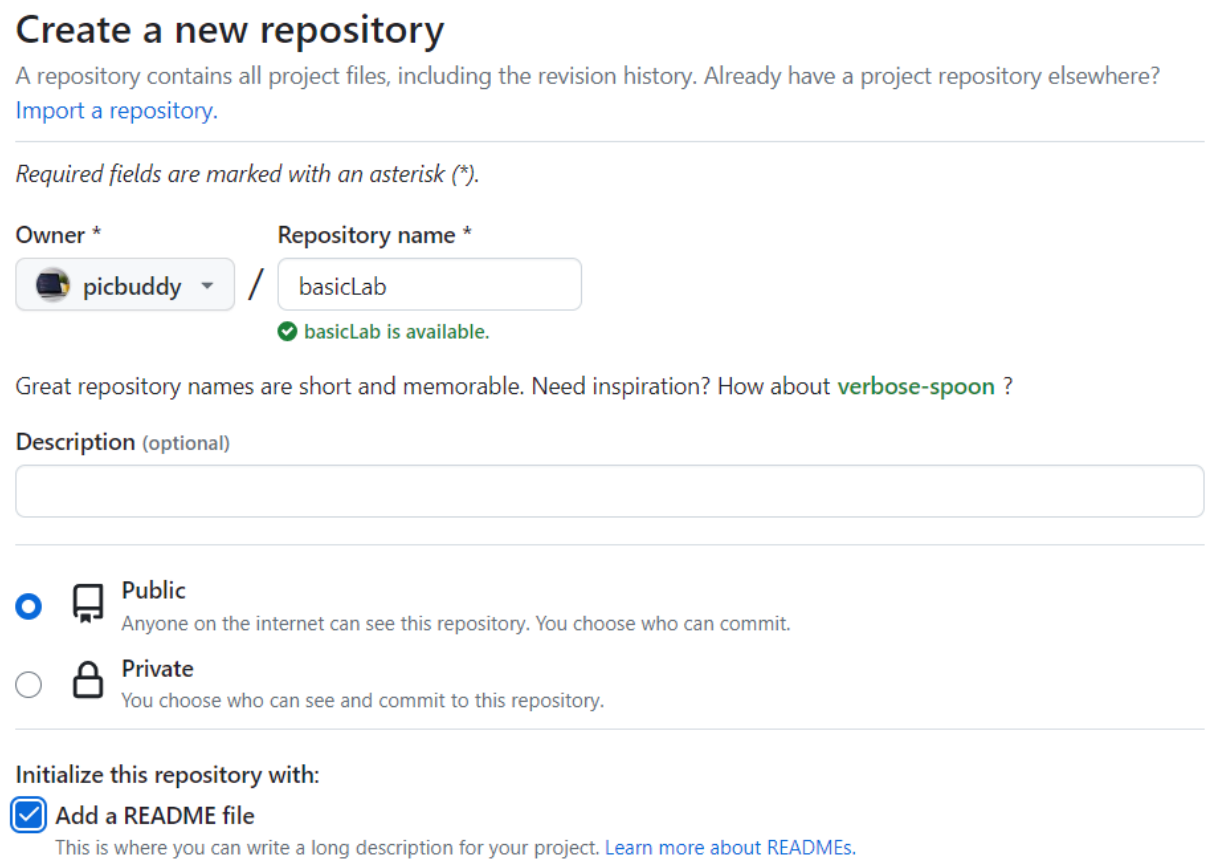
작업폴더생성 후 Terminal(PowerShell) 사용

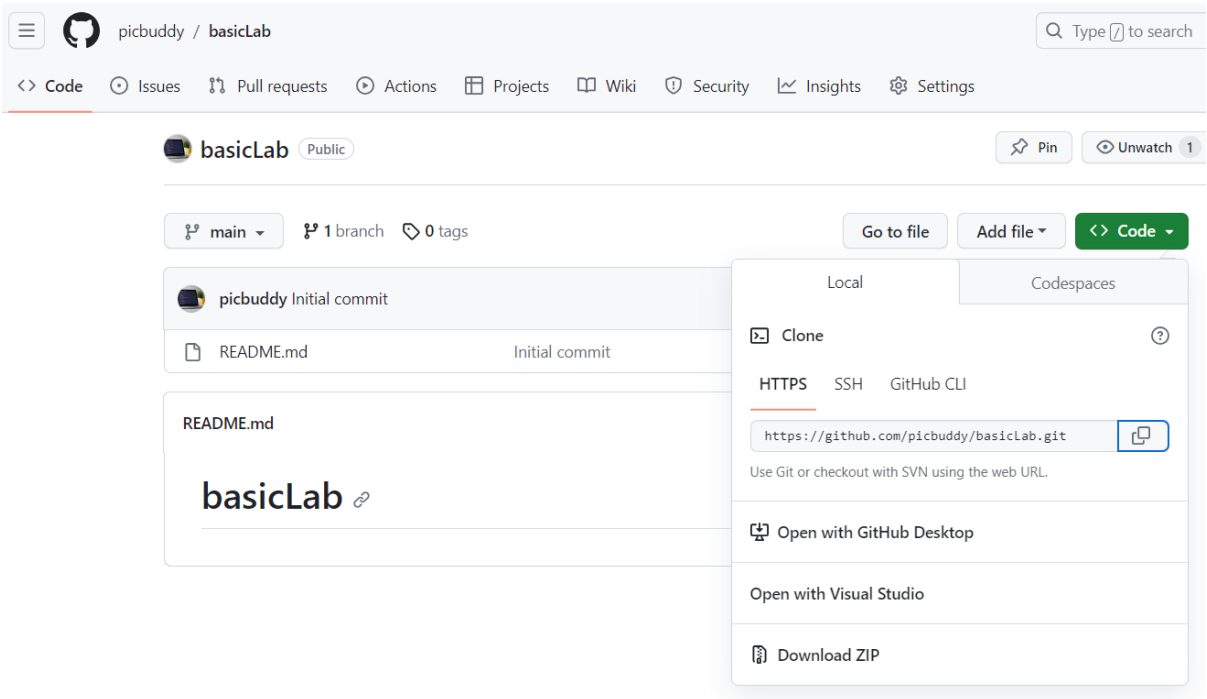
- **echo "# test2" >> README.md**
- **git init (git initialize : .git 폴더 생성)**
- **git add README.md (파일을 stage 영역에 올린다)**
- **git commit -m "first commit" (git history 작성, -m: 인라인 커밋 메시지 작성)**
- **git branch -M main (-M : 브랜치명 강제 변경)**
- **git remote add origin https://github.com/picbuddy/basicLab.git**
- **git push -u origin main (-u : 다음부터는 git push 만 써도 됨)**


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows
PS C:\basicLab> echo "# basicLab" >> README.md
PS C:\basicLab> git init
Initialized empty Git repository in C:/basicLab/.git/
PS C:\basicLab> git add READ<E.md
fatal: pathspec 'READ<E.md' did not match any files
PS C:\basicLab> git add README.md
PS C:\basicLab> git commit -m "first commit"
[master (root-commit) 1a1decf] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
PS C:\basicLab> git branch -M main
PS C:\basicLab> git remote add origin https://github.com/picbuddy/basicLab.git
PS C:\basicLab> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 231 bytes | 231.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/picbuddy/basicLab.git
* [new branch] main -> main
branch 'main' set up to track 'origin/main'.
PS C:\basicLab>

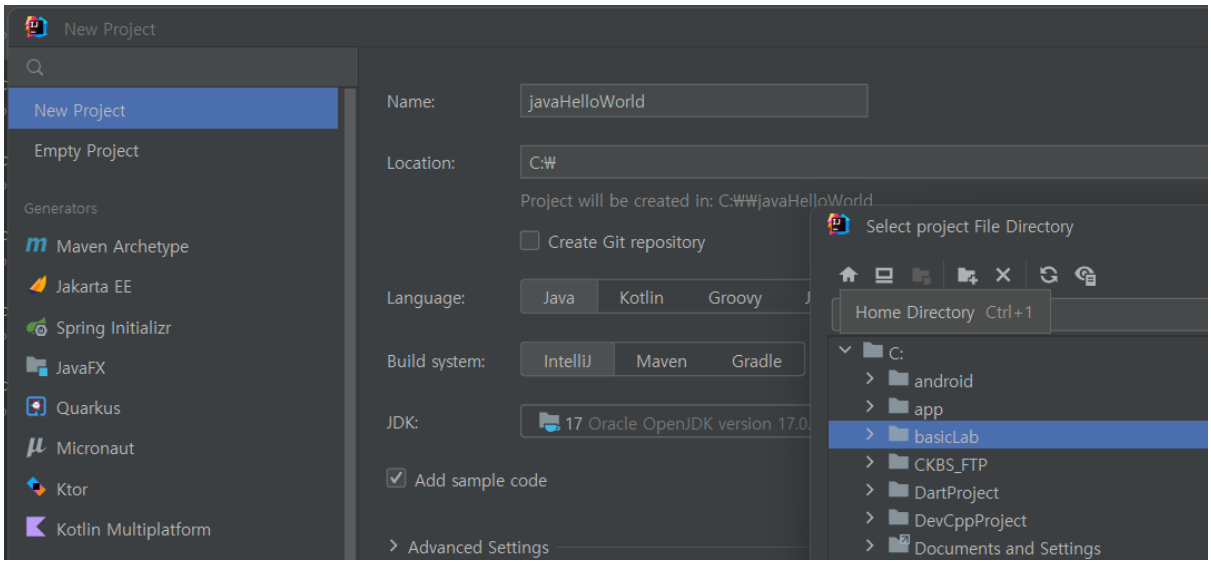
Git Initialize 포함하여 Repository 생성 → Clone 후 IntelliJ 프로젝트 commit 해보기

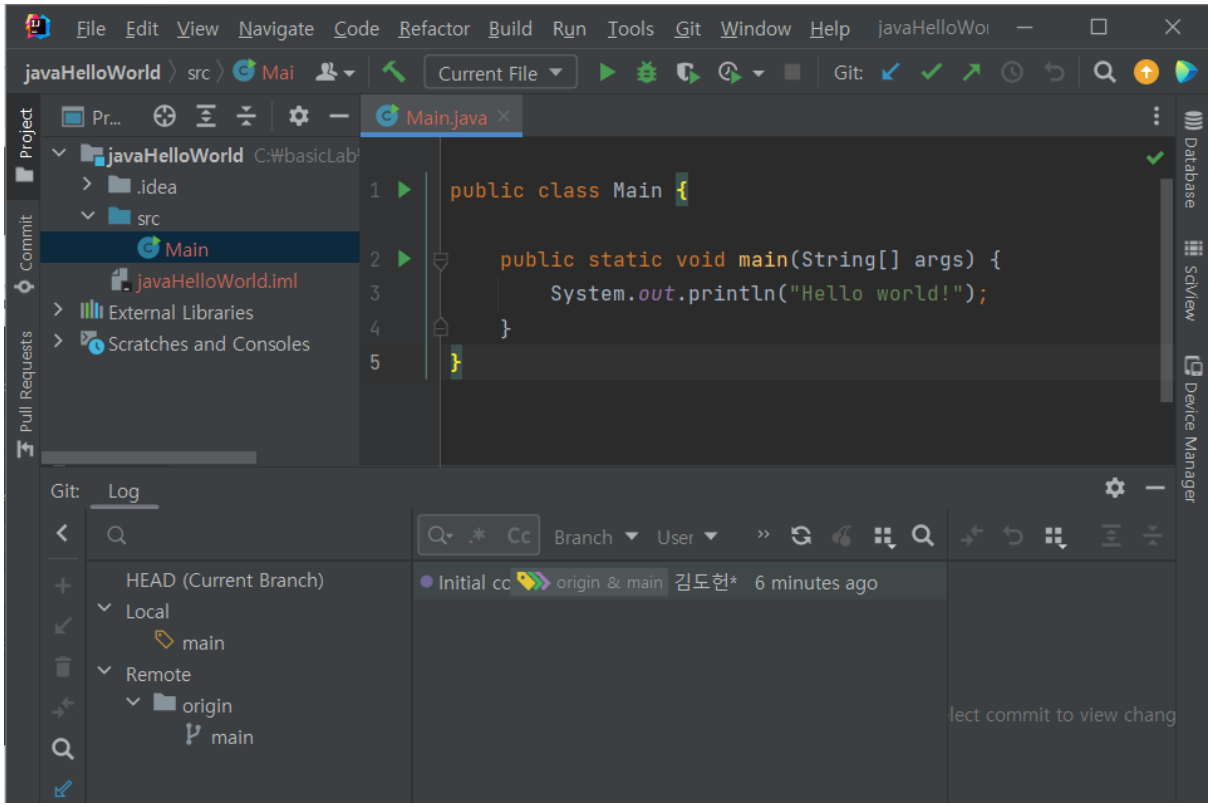
git clone "URL"

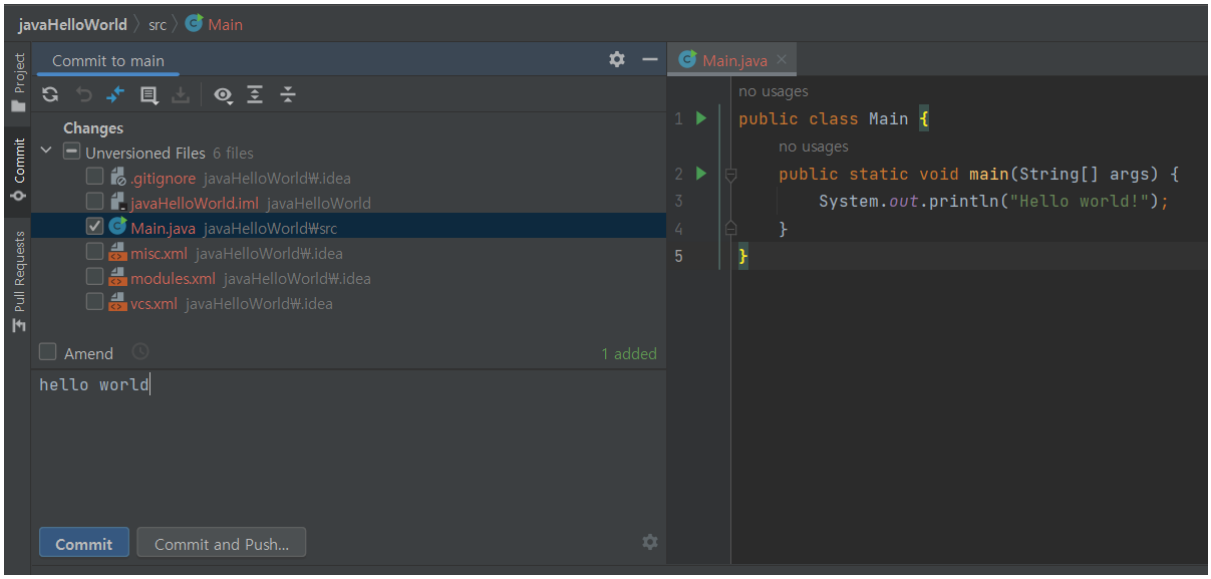


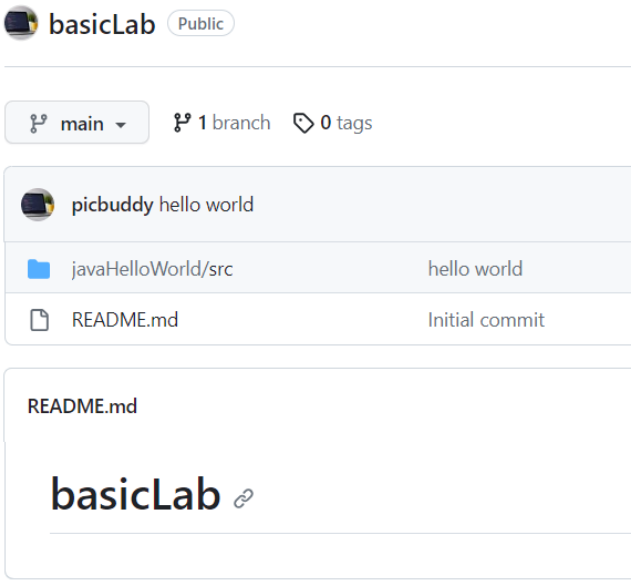


```
PS C:\> git clone https://github.com/picbuddy/basicLab.git
Cloning into 'basicLab'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
PS C:\> cd basicLab
PS C:\basicLab>
```







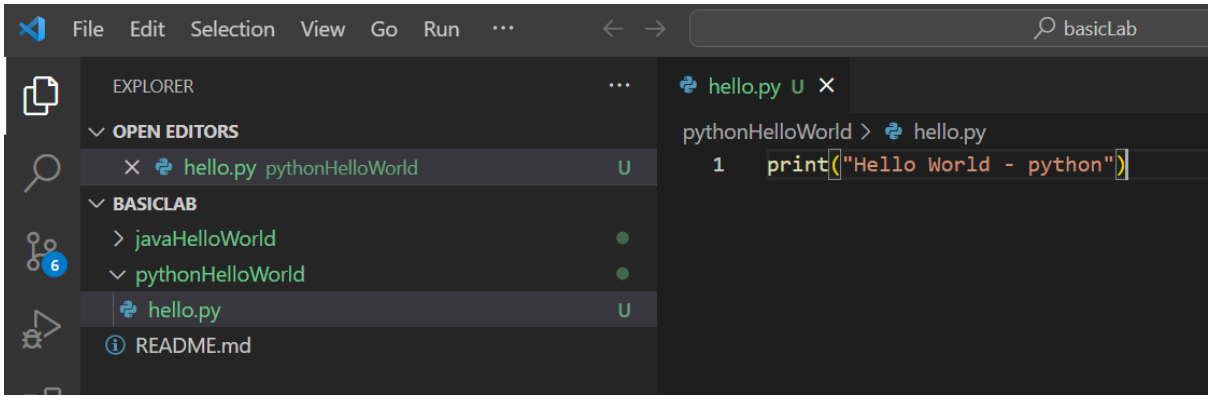


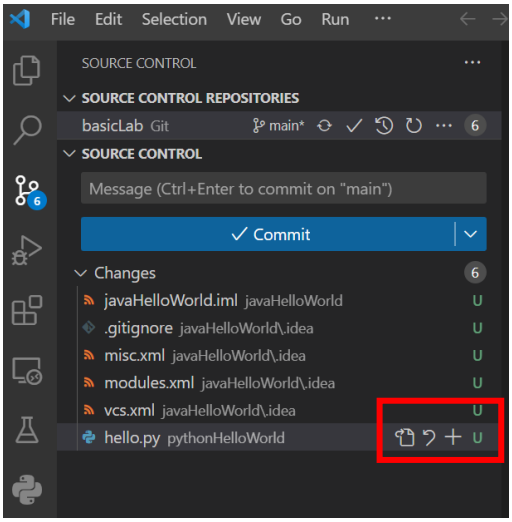
CLI : update my local repo to remote repo

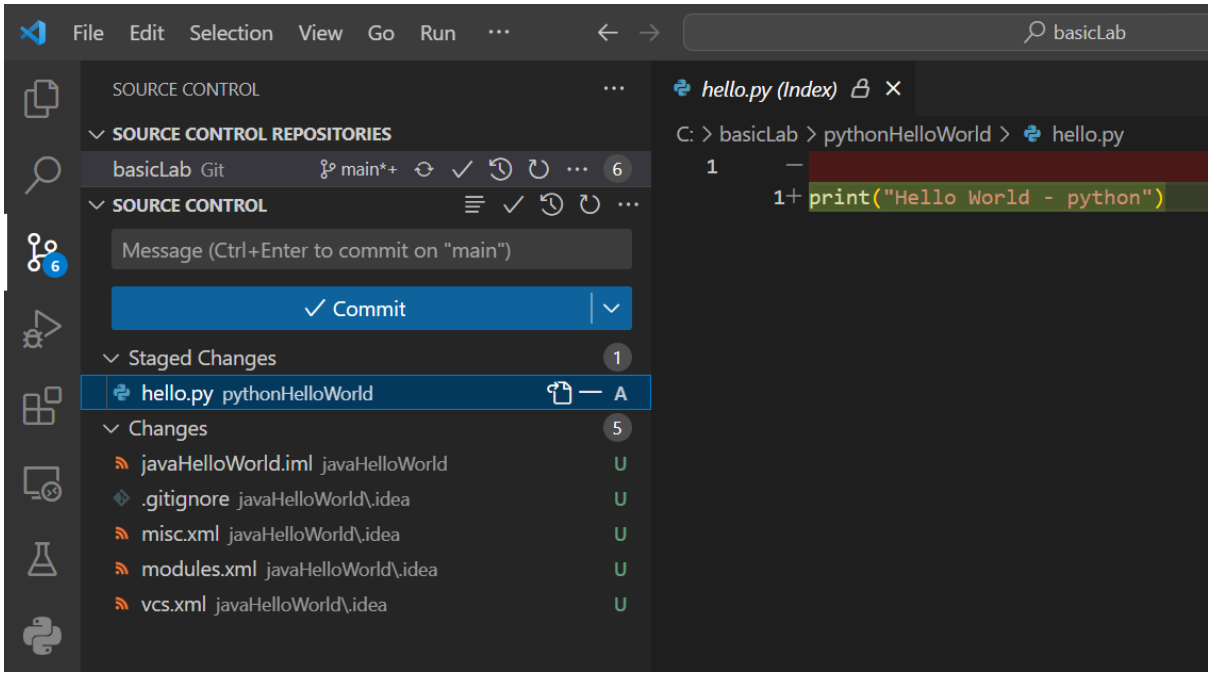
1. 추가할 파일 더하기 : git add Main.java
2. 히스토리 만들기 : git commit -m "comment"
3. Github 로 올리기 : git push origin main

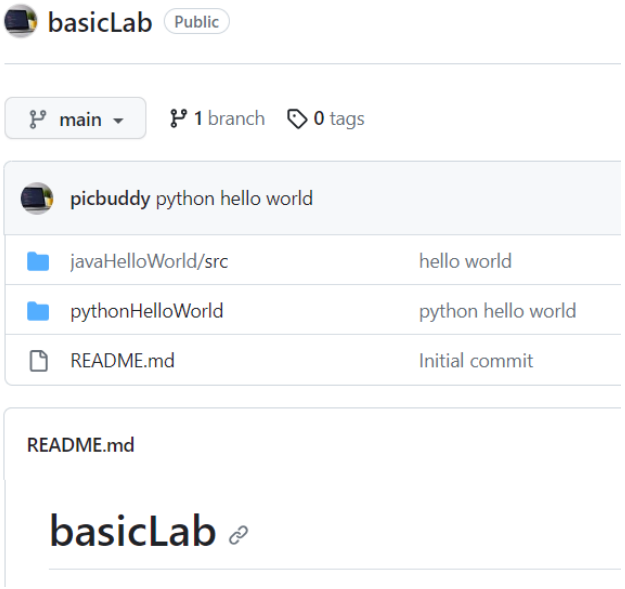
```
PS C:\basicLab\HelloWorld\src> git branch
* main
PS C:\basicLab\HelloWorld\src> git add Main.java
PS C:\basicLab\HelloWorld\src> git commit -m "add main.java"
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../.idea/
        ../HelloWorld.iml
        ../out/

nothing added to commit but untracked files present (use "git add" to track)
PS C:\basicLab\HelloWorld\src> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 750 bytes | 750.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/picbuddy/basicLab.git
   bf8c910..880f858  main -> main
PS C:\basicLab\HelloWorld\src>
```

Git Initialize 포함하여 Repository 생성 → Clone 후 VS CODE 프로젝트 commit 해보기

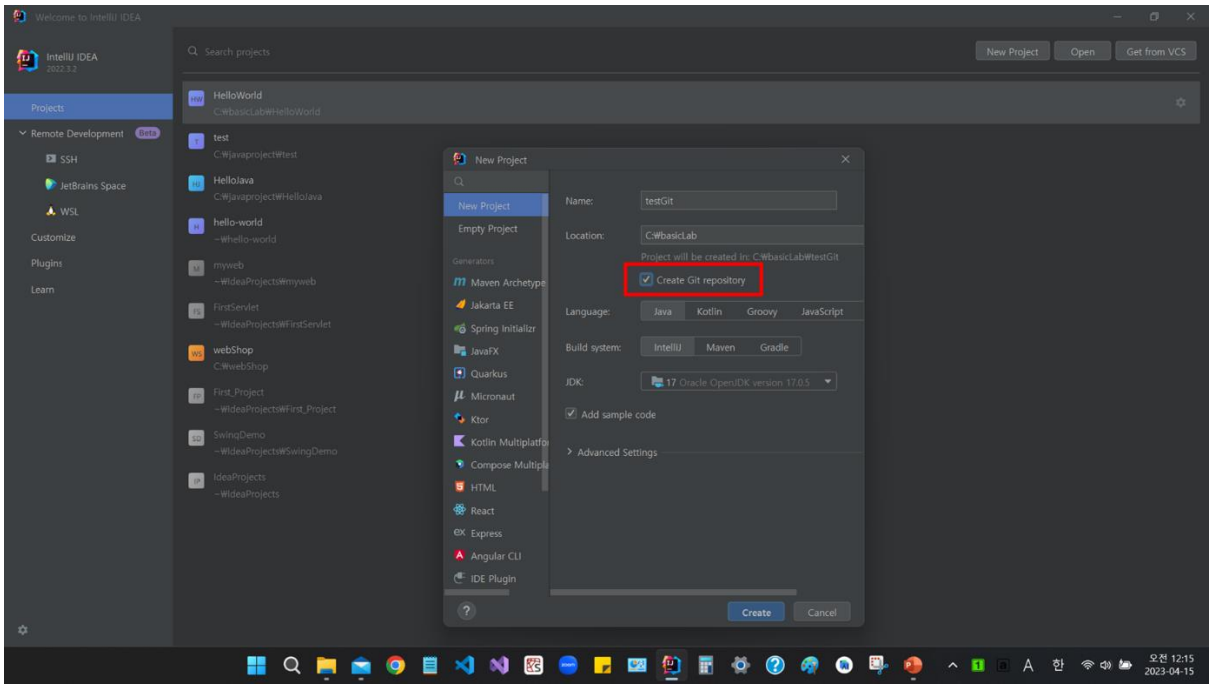


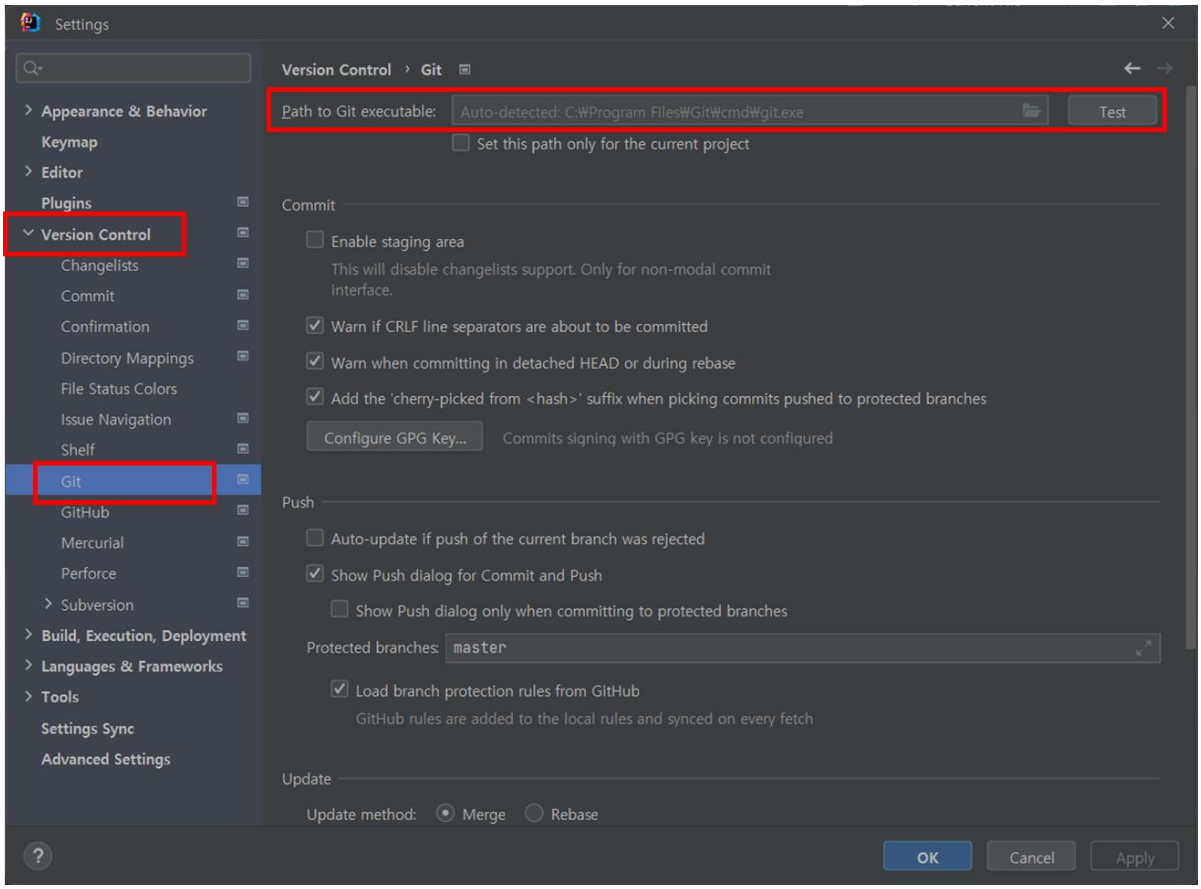


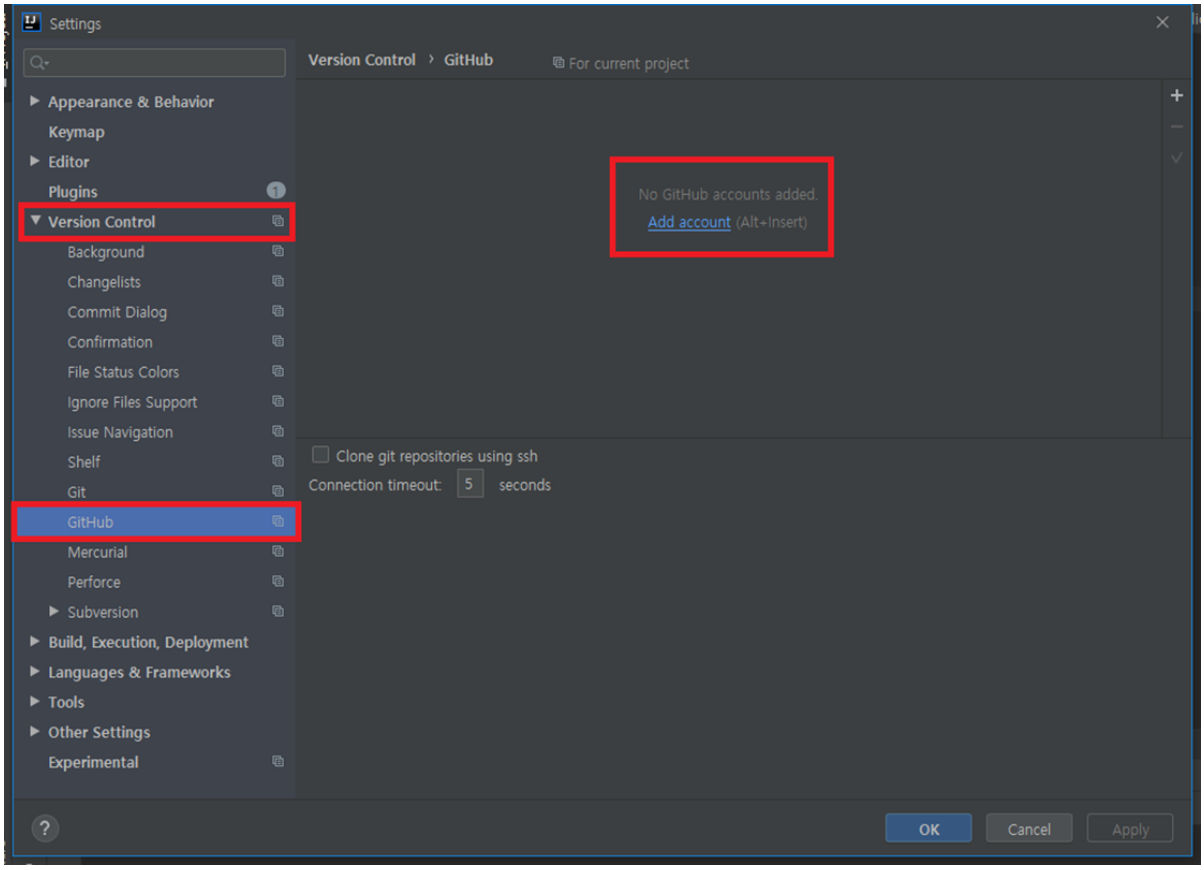

File Edit Selection View Go Run
basicLab
SOURCE CONTROL
SOURCE CONTROL REPOSITORIES
basicLab Git main*+ 6
SOURCE CONTROL
Message (Ctrl+Enter to commit on "main")
Commit
Staged Changes 1
hello.py pythonHelloWorld A
Changes 5
javaHelloWorld.iml javaHelloWorld U
.gitignore javaHelloWorld\.idea U
misc.xml javaHelloWorld\.idea U
modules.xml javaHelloWorld\.idea U
vcs.xml javaHelloWorld\.idea U
hello.py (Index)
C: > basicLab > pythonHelloWorld > hello.py
1 -
1+ print("Hello World - python")

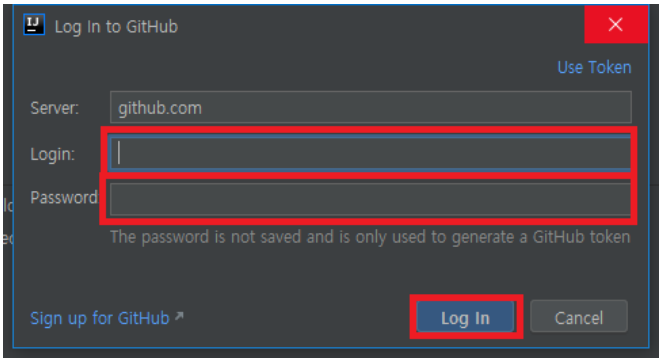

basicLab
Public
main
1 branch
0 tags
picbuddy python hello world
javaHelloWorld/src hello world
pythonHelloWorld python hello world
README.md Initial commit
README.md
basicLab

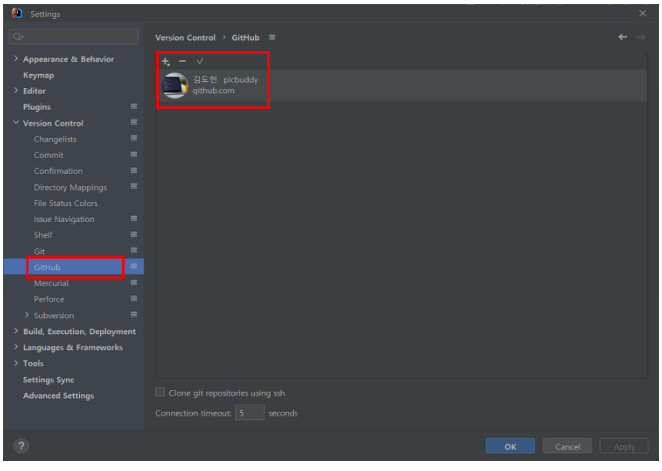
APPENDIX : IntelliJ 에서 Git Repository 같이 생성하기

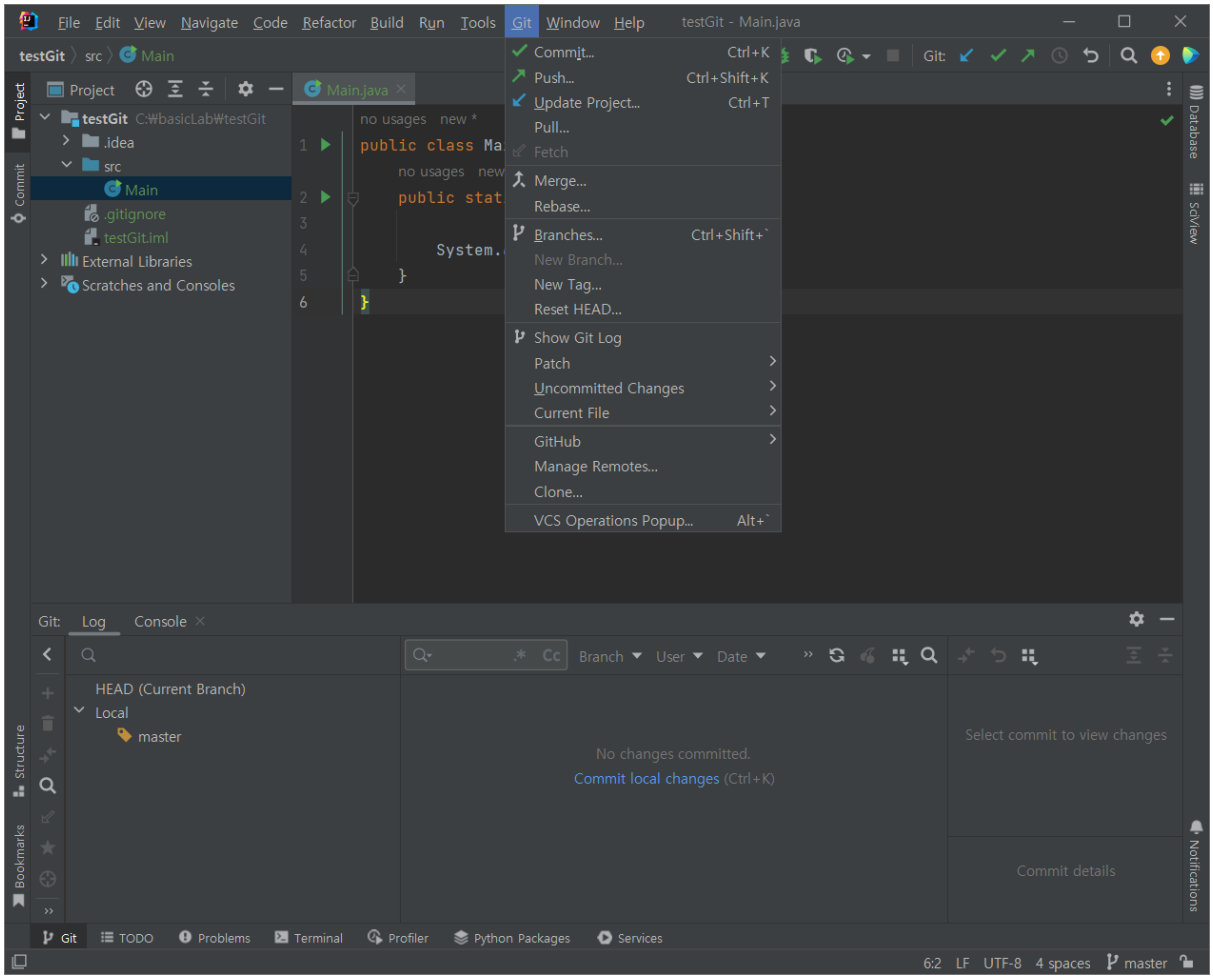





Settings
Version Control › GitHub
For current project
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Background
Changelists
Commit Dialog
Confirmation
File Status Colors
Ignore Files Support
Issue Navigation
Shelf
Git
GitHub
Mercurial
Perforce
Subversion
Build, Execution, Deployment
Languages & Frameworks
Tools
Other Settings
Experimental
No GitHub accounts added.
Add account (Alt+Insert)
Clone git repositories using ssh
Connection timeout: 5 seconds
OK
Cancel
Apply

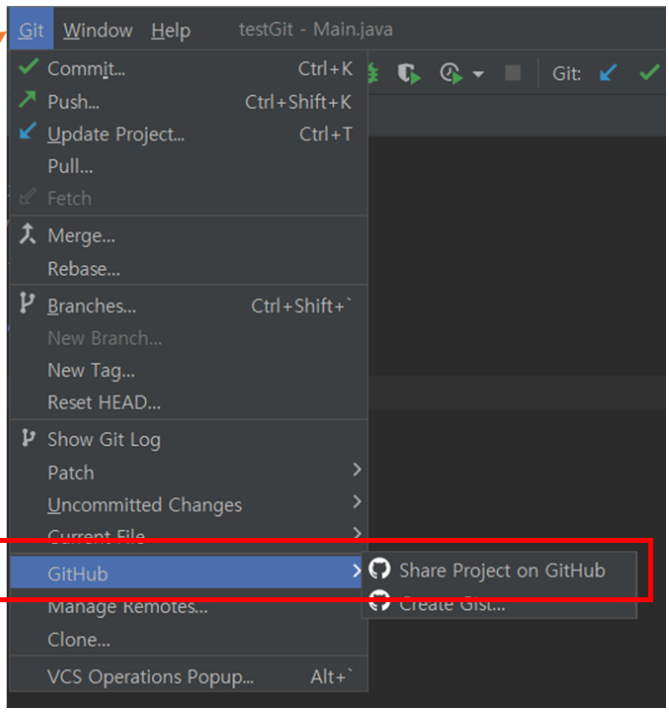

Log In to GitHub
Use Token
Server: github.com
Login:
Password
The password is not saved and is only used to generate a GitHub token
Sign up for GitHub
Log In
Cancel

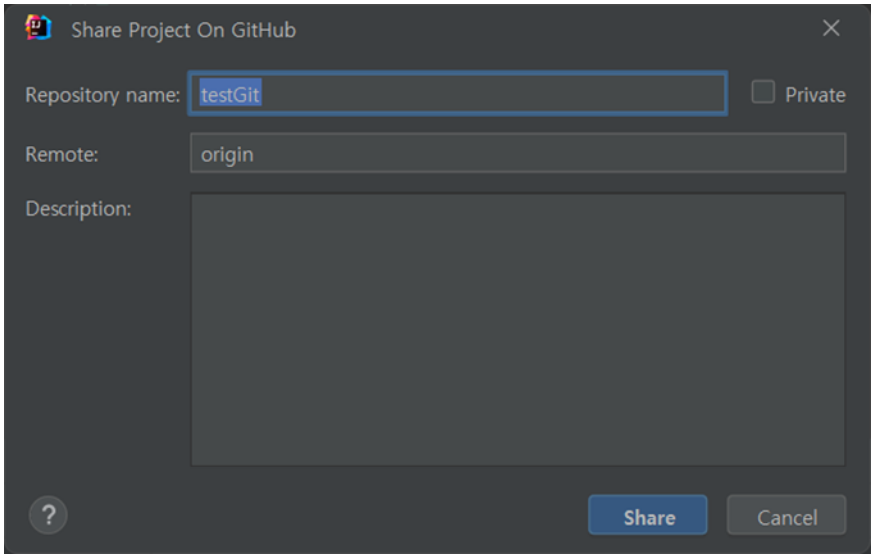


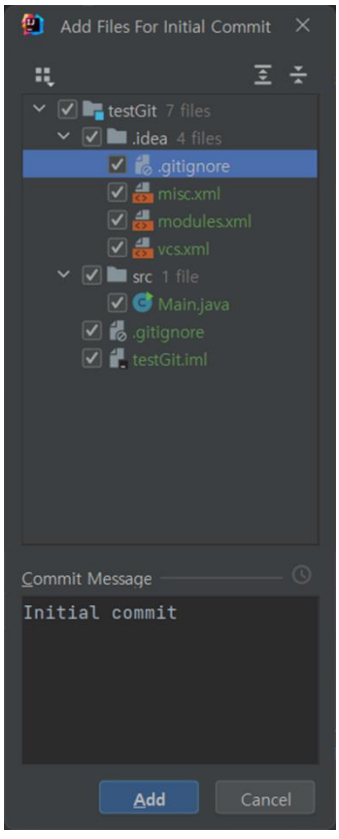


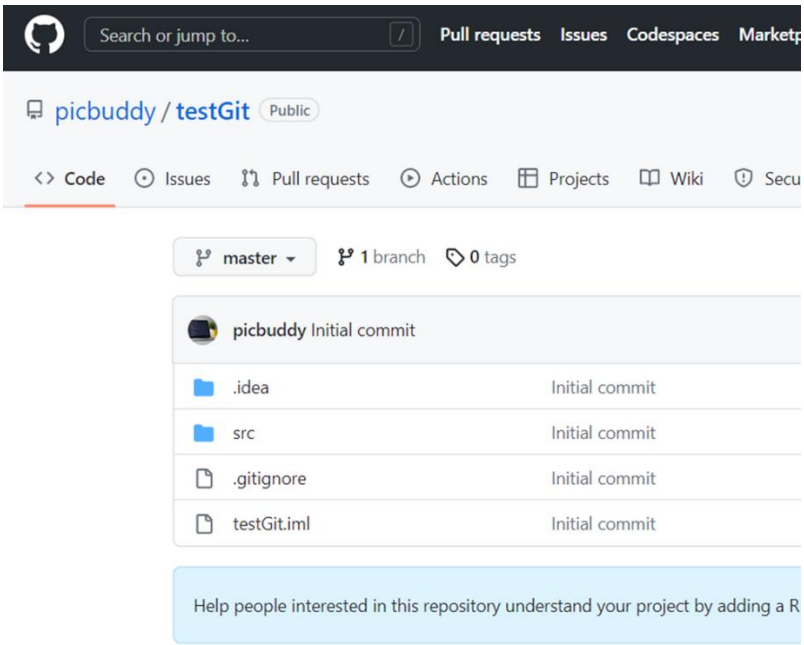
프로젝트 명으로 자동으로 Repository 를 생성하여 Github 연동

프로젝트 폴더에 Local Git(.git)이 없다면 VCS Local Git(.git) 있다면 Git으로 표시




Share Project On GitHub
Repository name:
testGit
Private
Remote:
origin
Description:
Share
Cancel

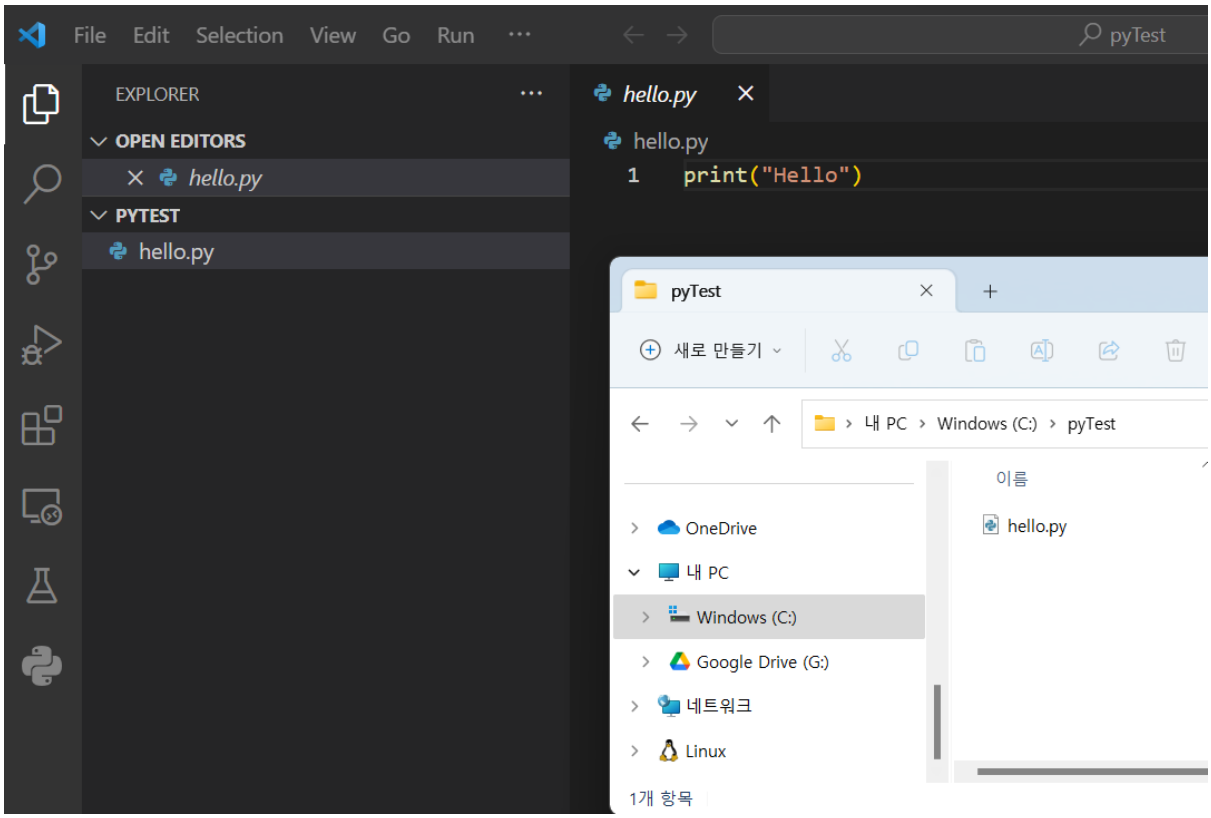

Add Files For Initial Commit
testGit 7 files
.idea 4 files
.gitignore
misc.xml
modules.xml
vcs.xml
src 1 file
Main.java
.gitignore
testGit.iml
Commit Message
Initial commit
Add
Cancel


Search or jump to...
Pull requests
Issues
Codespaces
picbuddy / testGit
Public
Code
Issues
Pull requests
Actions
Projects
Wiki
master
1 branch
0 tags
picbuddy Initial commit
.idea
Initial commit
src
Initial commit
.gitignore
Initial commit
testGit.iml
Initial commit
Help people interested in this repository understand your project by adding a R

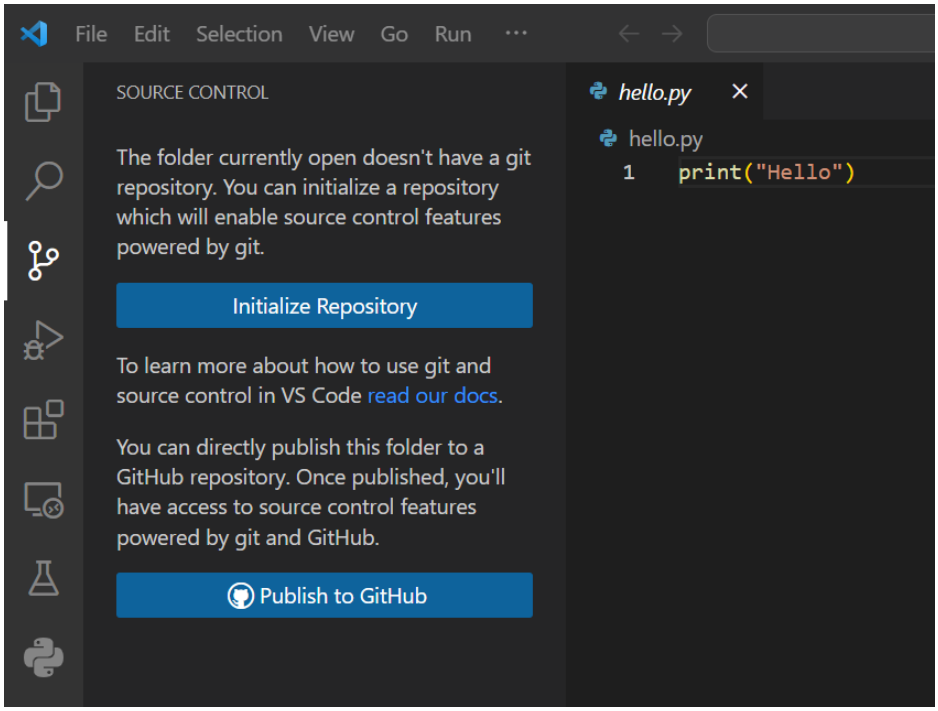
VSCODE 에서 Git Repositiory 만들기

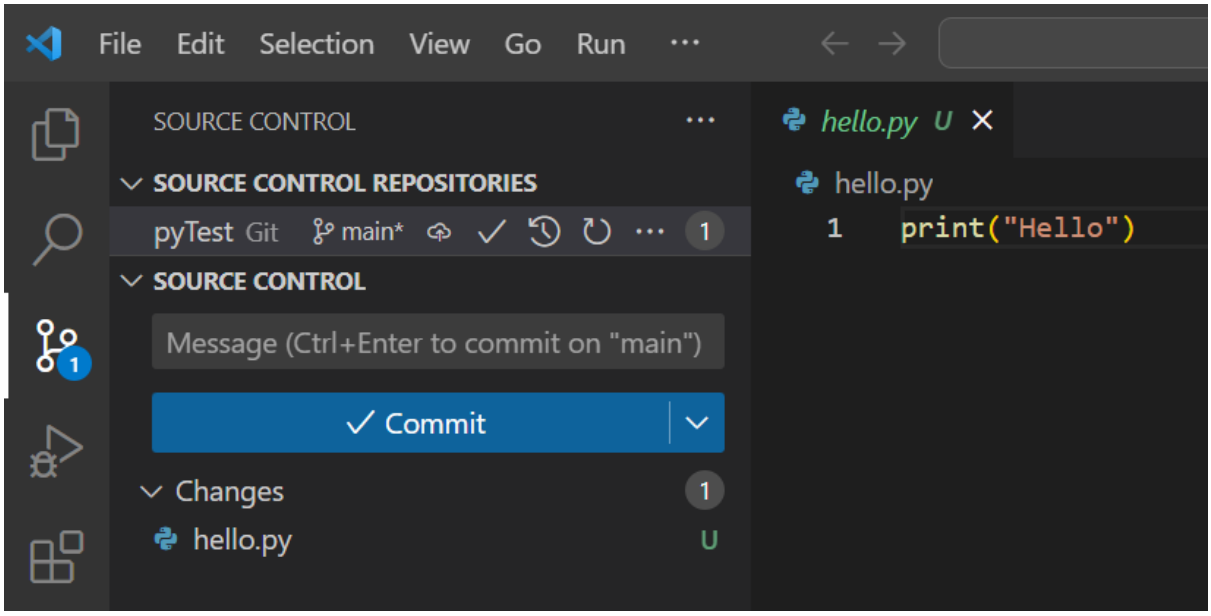
- **폴더 생성 후 프로젝트 만들기**

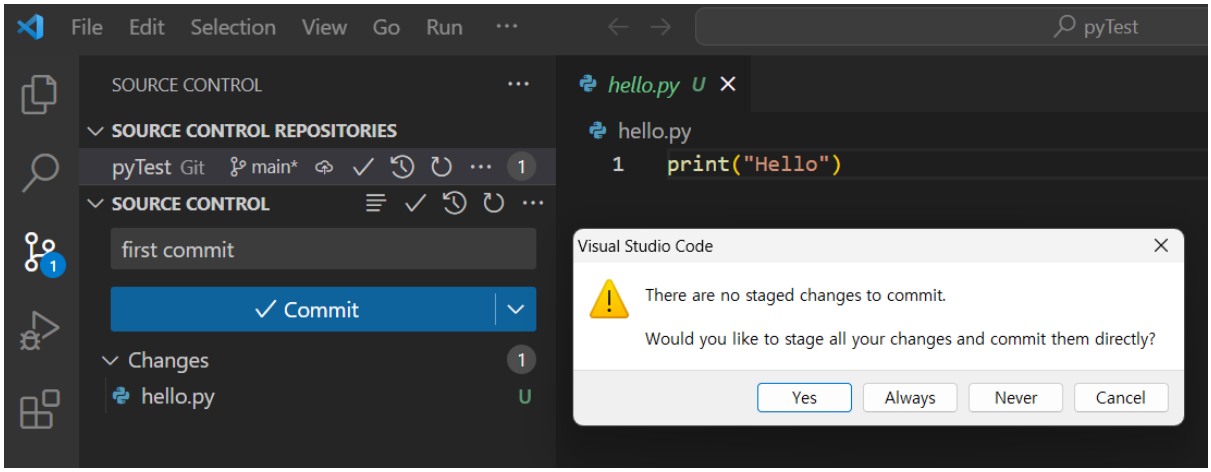
New file → Python File → 폴더 생성 후 저장



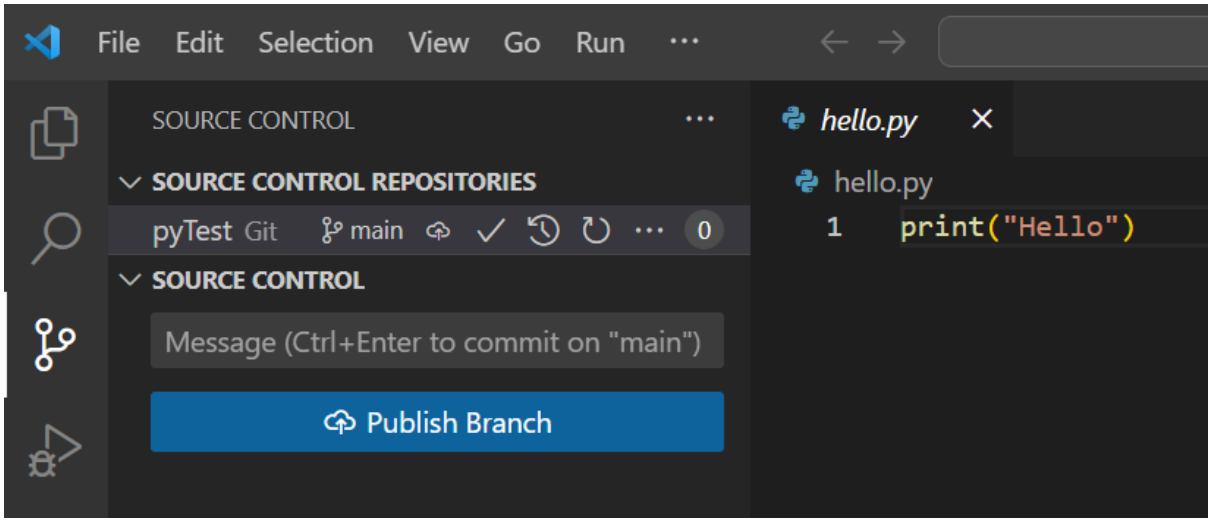
Initialize Repository

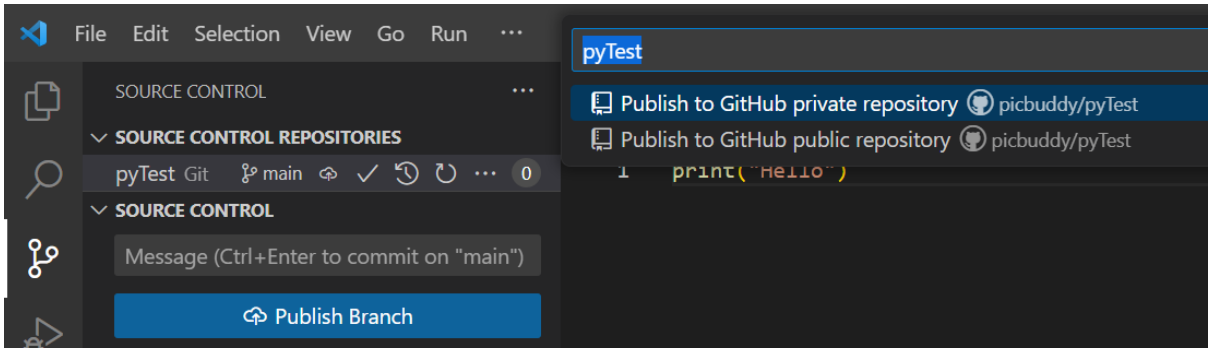


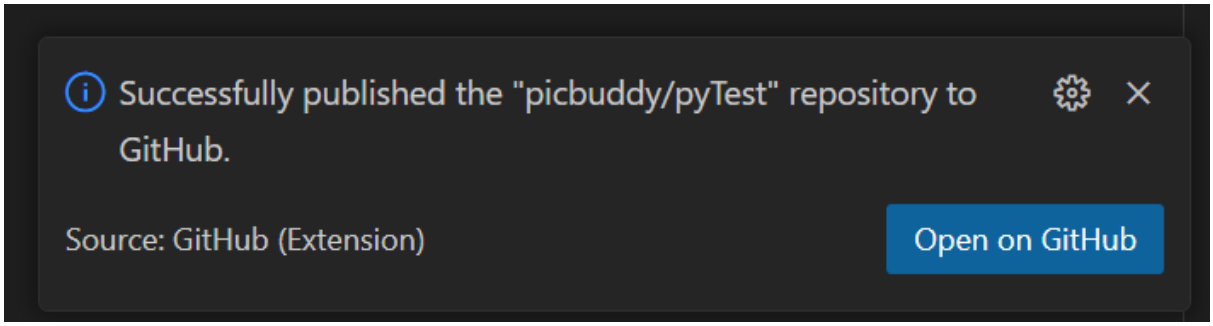


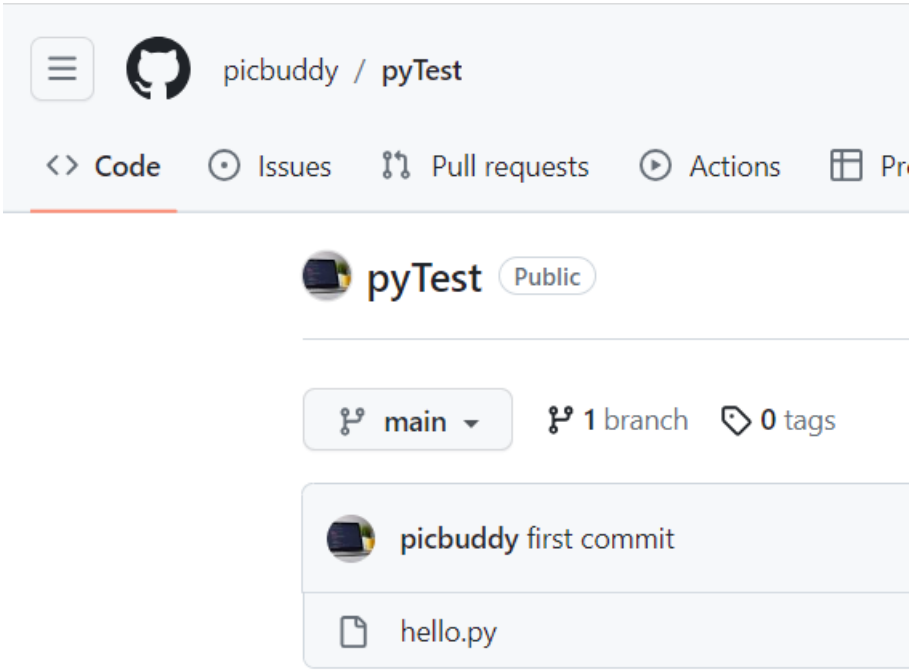


Publish to GitHub



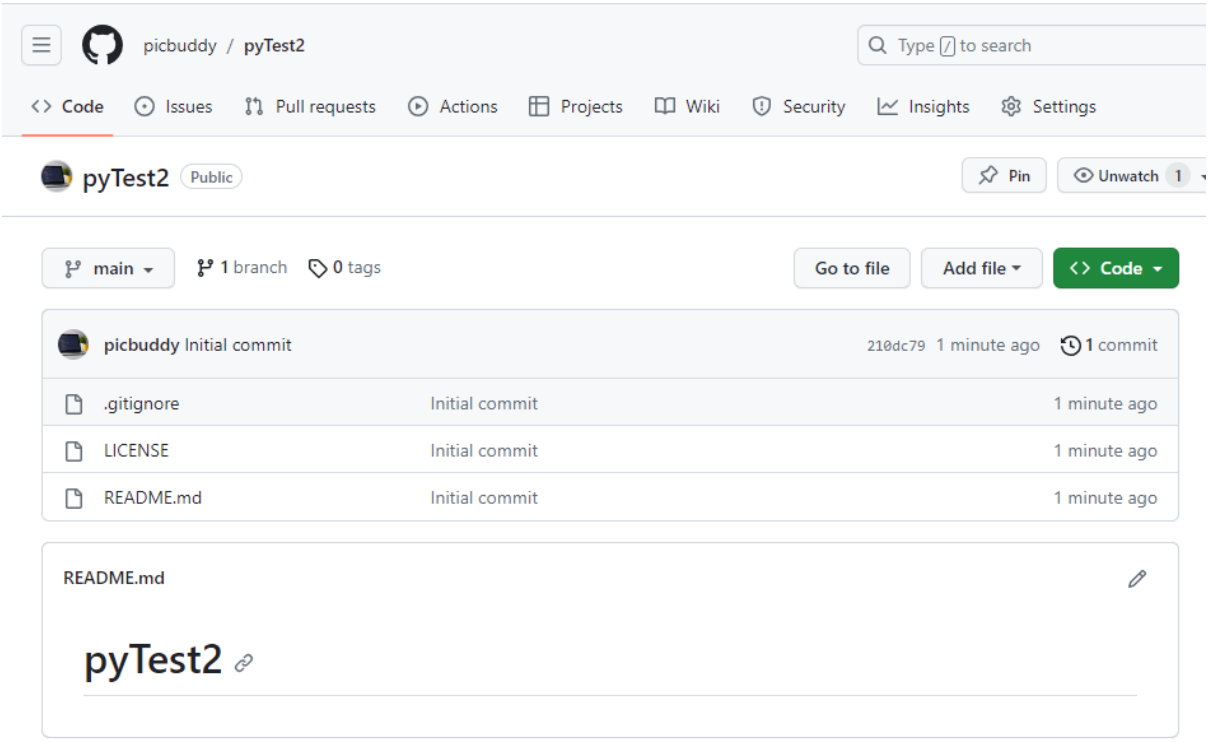

File Edit Selection View Go Run
SOURCE CONTROL
SOURCE CONTROL REPOSITORIES
pyTest Git main 0
SOURCE CONTROL
Message (Ctrl+Enter to commit on "main")
Publish Branch
pyTest
Publish to GitHub private repository picbuddy/pyTest
Publish to GitHub public repository picbuddy/pyTest
print("Hello")


Successfully published the "picbuddy/pyTest" repository to GitHub.
Source: GitHub (Extension)
Open on GitHub

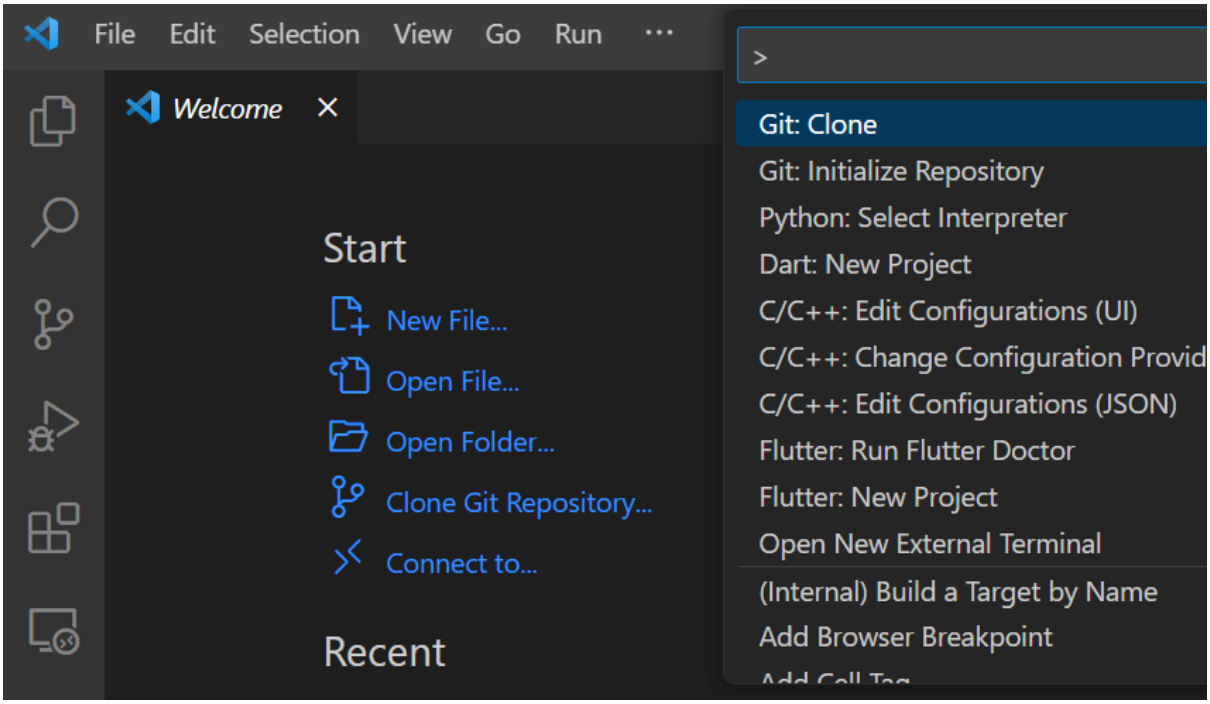

picbuddy / pyTest
Code
Issues
Pull requests
Actions
pyTest
Public
main
1 branch
0 tags
picbuddy first commit
hello.py

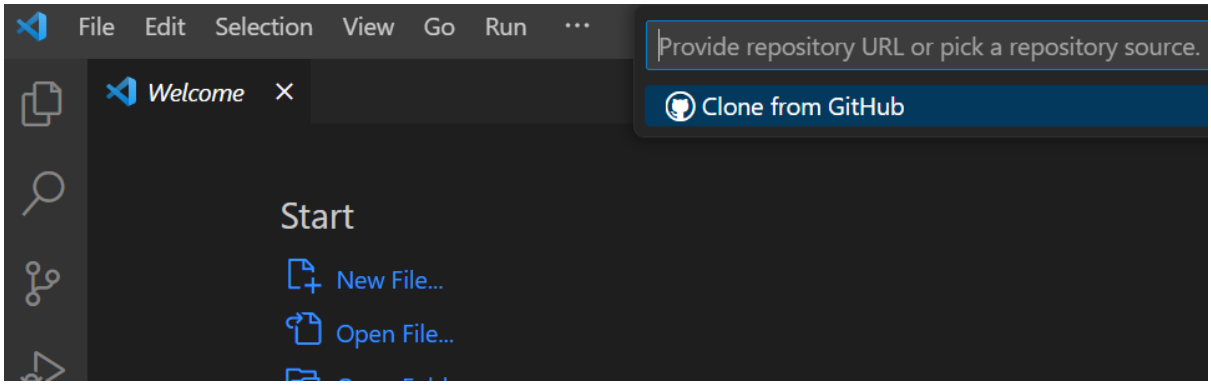
VSCODE 로 GitHub 에서 생성된 Repository 로 파일 업로드 하기

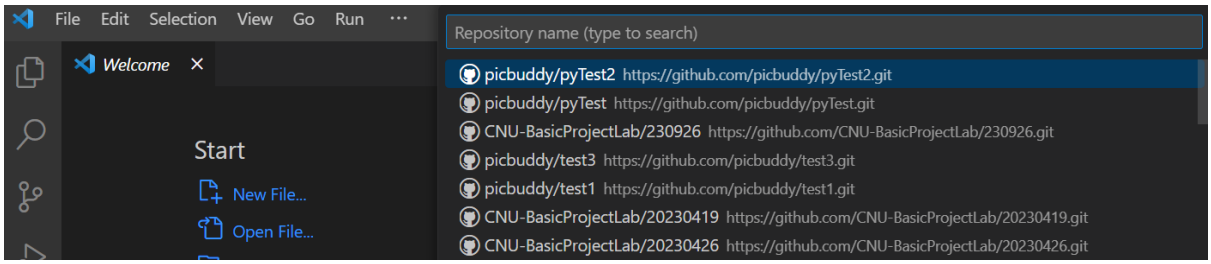
GitHub 에서 Repository 생성 후

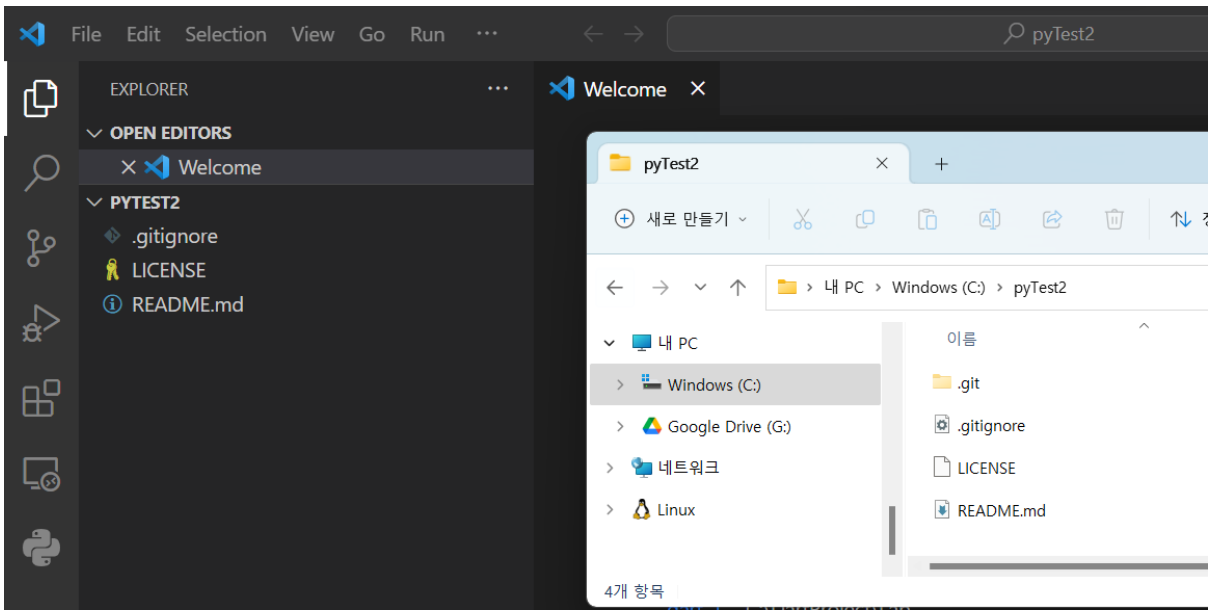


VSCODE 에서 F1 → git Clone 후 코드 작성

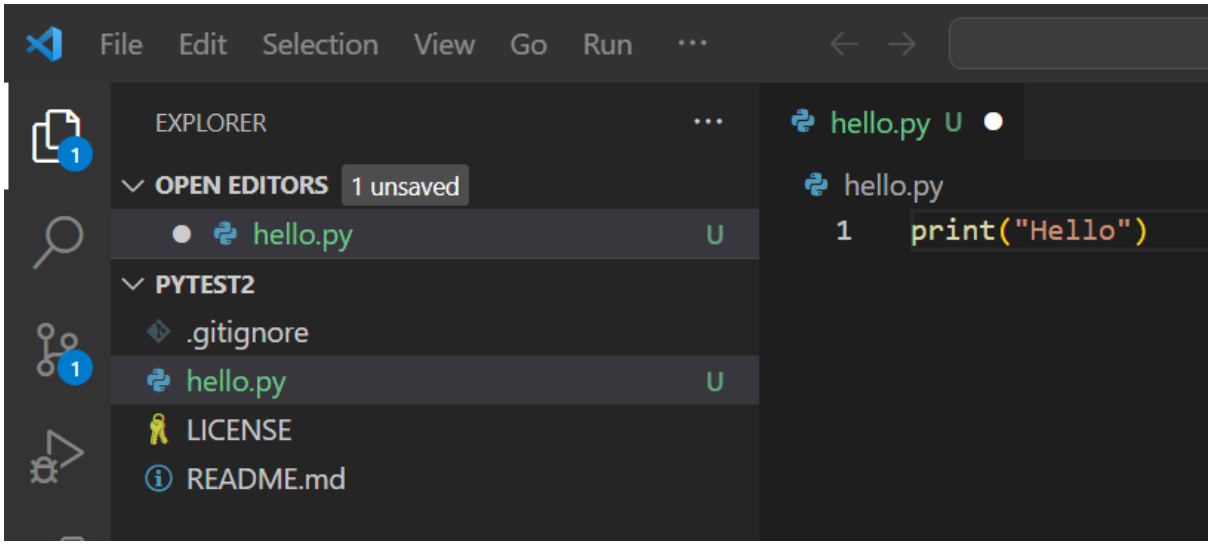


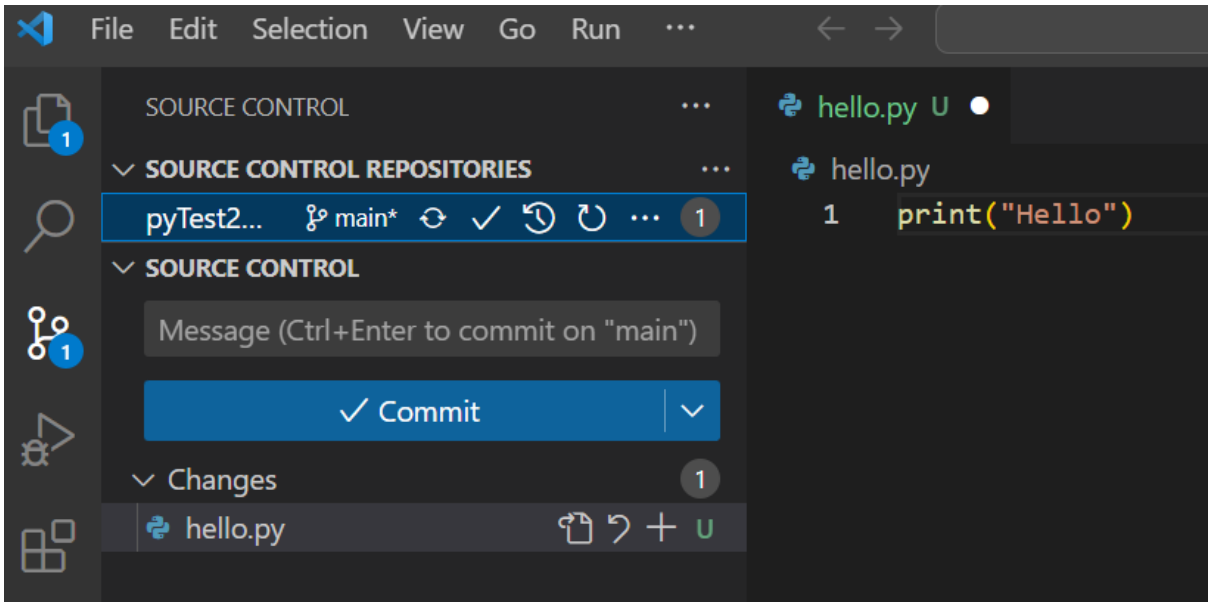

File Edit Selection View Go Run ···
Welcome
Provide repository URL or pick a repository source.
Clone from GitHub
Start
New File...
Open File...

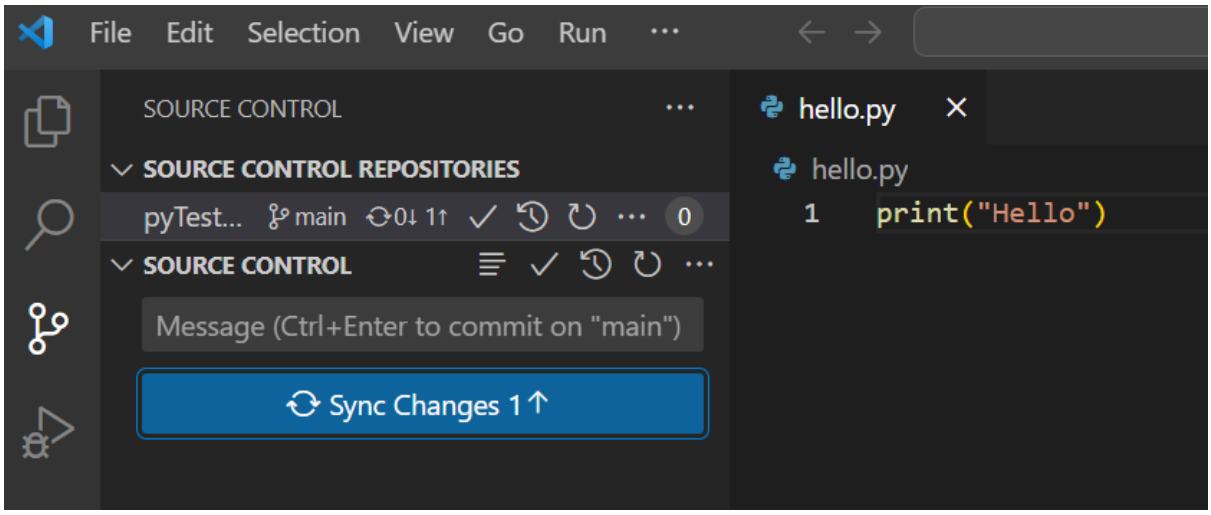

File Edit Selection View Go Run ···
Welcome
Repository name (type to search)
picbuddy/pyTest2 https://github.com/picbuddy/pyTest2.git
picbuddy/pyTest https://github.com/picbuddy/pyTest.git
CNU-BasicProjectLab/230926 https://github.com/CNU-BasicProjectLab/230926.git
picbuddy/test3 https://github.com/picbuddy/test3.git
picbuddy/test1 https://github.com/picbuddy/test1.git
CNU-BasicProjectLab/20230419 https://github.com/CNU-BasicProjectLab/20230419.git
CNU-BasicProjectLab/20230426 https://github.com/CNU-BasicProjectLab/20230426.git
Start
New File...
Open File...

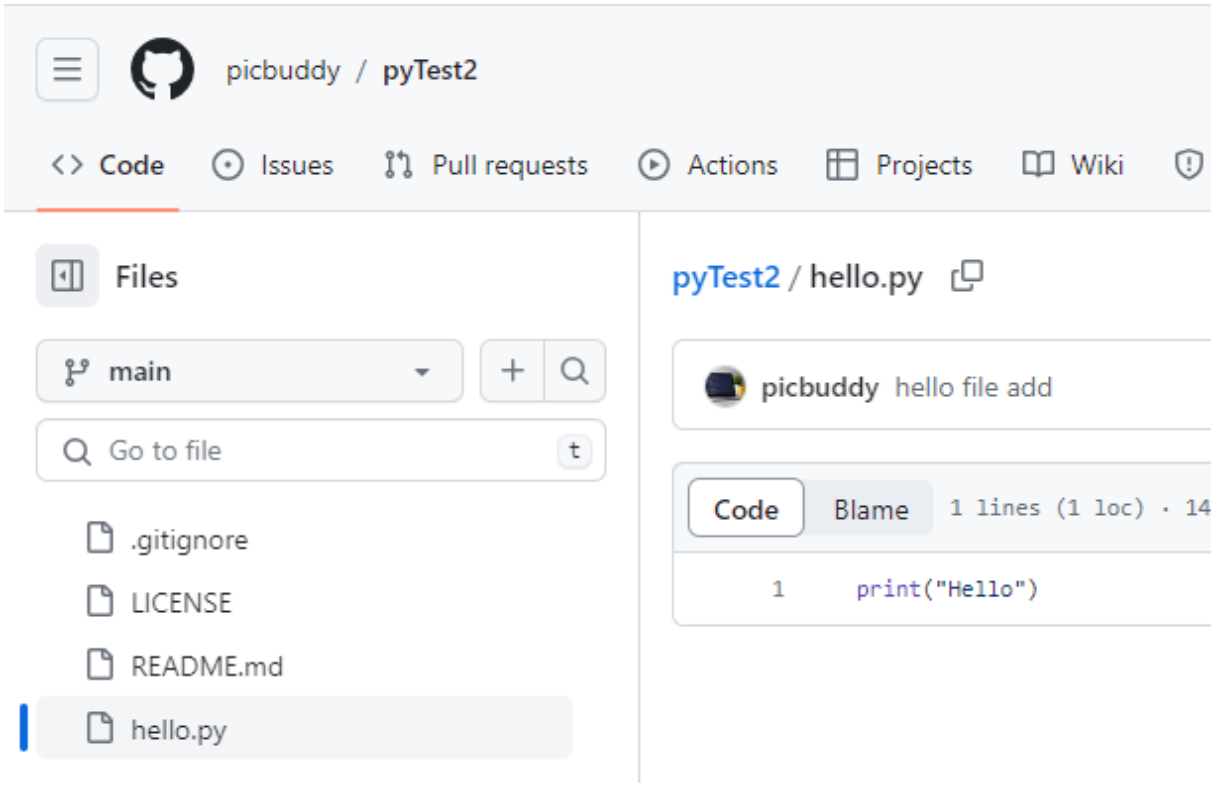

File Edit Selection View Go Run ···
pyTest2
EXPLORER
OPEN EDITORS
Welcome
PYTEST2
.gitignore
LICENSE
README.md
Welcome
pyTest2
새로 만들기
내 PC › Windows (C:) › pyTest2
내 PC
Windows (C:)
Google Drive (G:)
네트워크
Linux
이름
.git
.gitignore
LICENSE
README.md
4개 항목

File 작성


File Edit Selection View Go Run
EXPLORER
OPEN EDITORS 1 unsaved
hello.py U
PYTEST2
.gitignore
hello.py U
LICENSE
README.md
hello.py U
hello.py
1 print("Hello")


File Edit Selection View Go Run
SOURCE CONTROL
SOURCE CONTROL REPOSITORIES
pyTest2... main*
SOURCE CONTROL
Message (Ctrl+Enter to commit on "main")
Commit
Changes 1
hello.py U
hello.py U
hello.py
1 print("Hello")

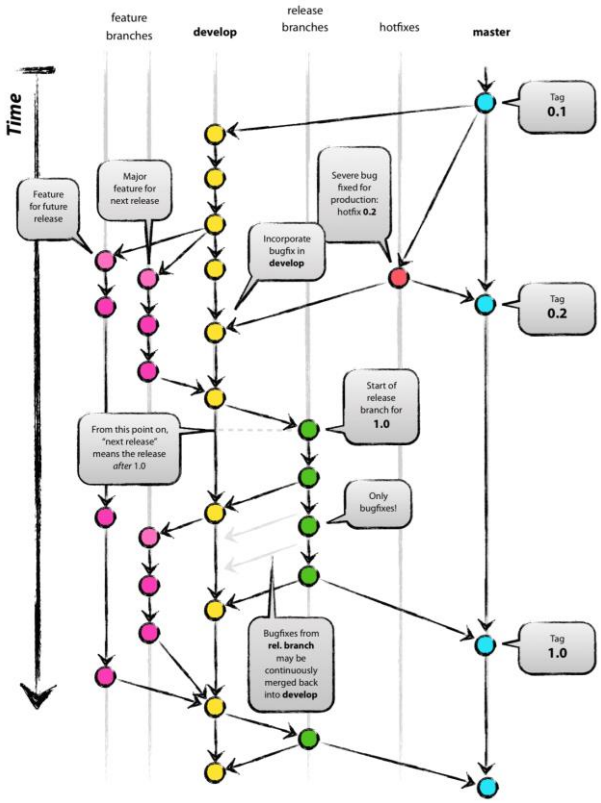

File Edit Selection View Go Run
SOURCE CONTROL
SOURCE CONTROL REPOSITORIES
pyTest... main 0↓ 1↑
SOURCE CONTROL
Message (Ctrl+Enter to commit on "main")
Sync Changes 1↑
hello.py
hello.py
1 print("Hello")


picbuddy / pyTest2
Code
Issues
Pull requests
Actions
Projects
Wiki
Files
main
Go to file
t
.gitignore
LICENSE
README.md
hello.py
pyTest2 / hello.py
picbuddy hello file add
Code
Blame
1 lines (1 loc) · 14
1 print("Hello")

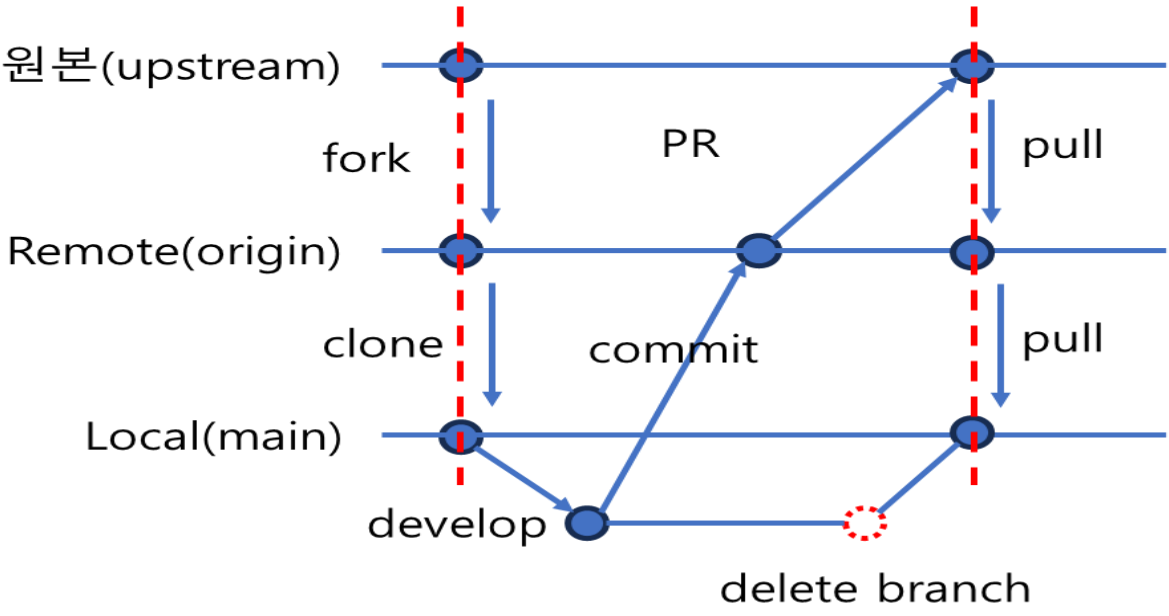
3-5. Work together with Github

APPENDIX : git branch strategy

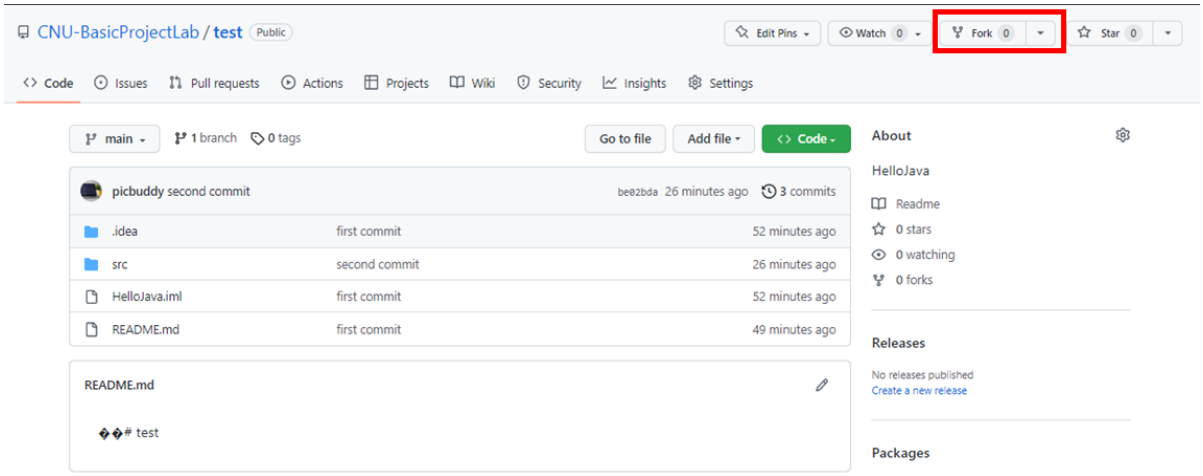
https://blog.dnd.ac/types-of-git-branch/

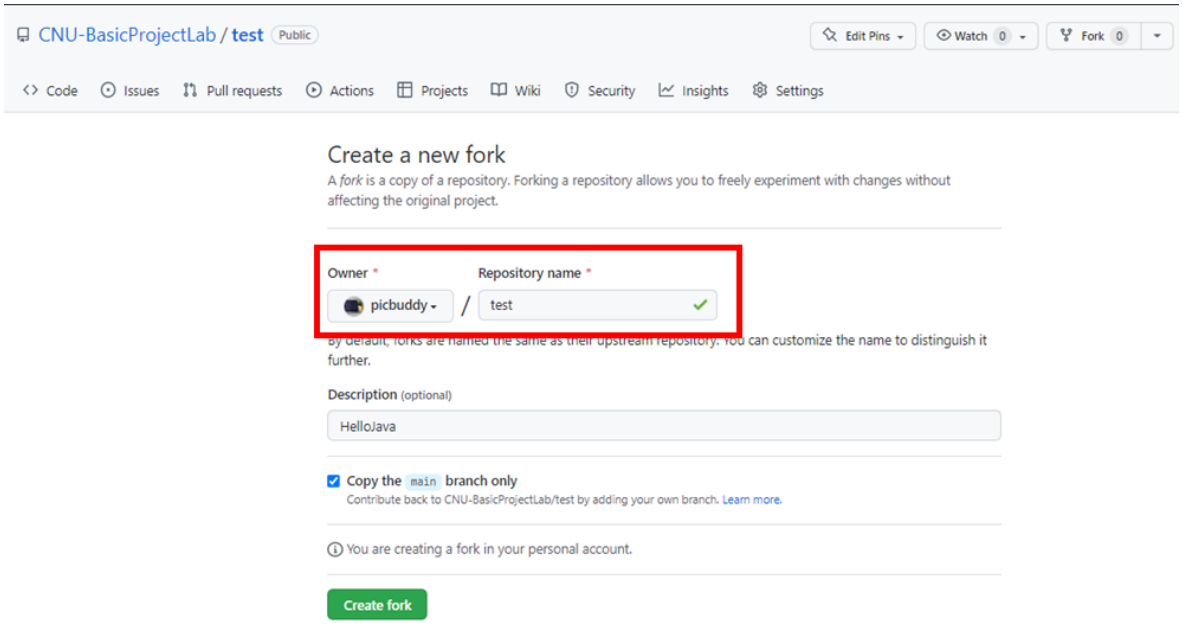


[실습]

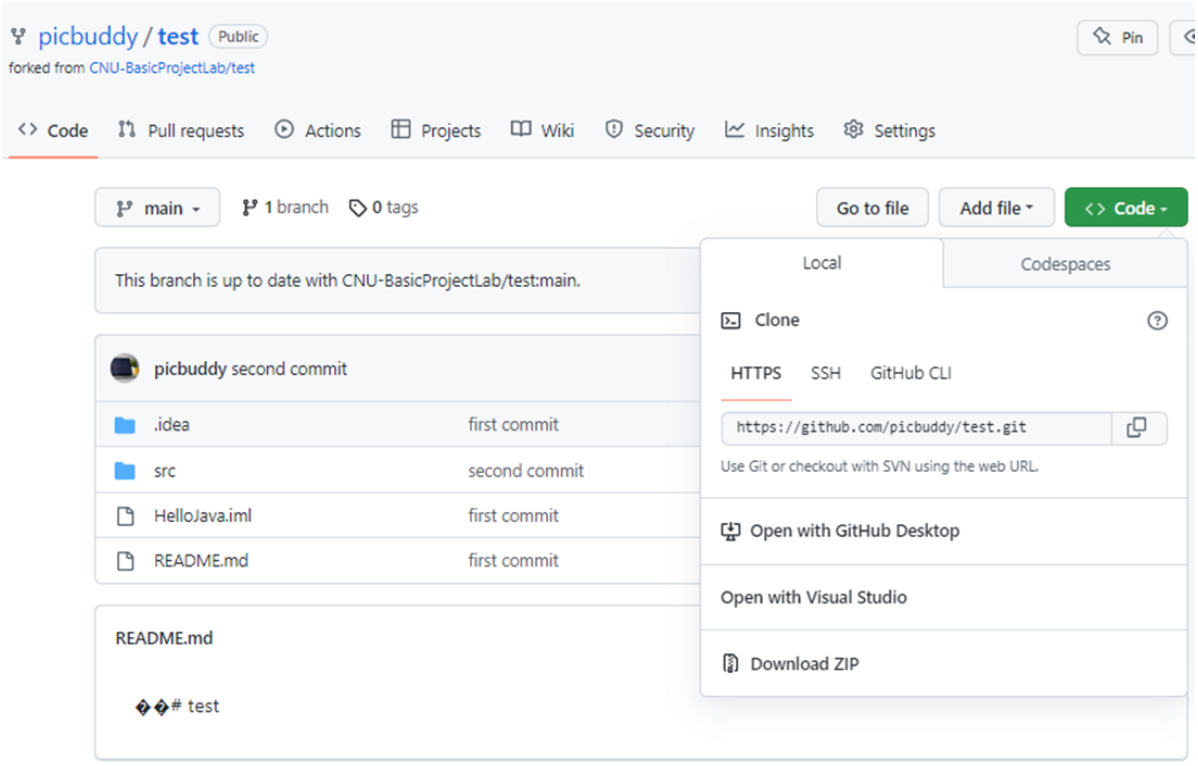


fork(my remote repo) → clone(my local repo) → branch(local) → commit & push(local) → pull(check from remote) → PR(pull request to owner)





fork(to my remote repo) → clone(my local repo) → branch(local) → commit & push(local) → pull(check remote) → PR(pull request to owner)



- Github 에서 소스코드 다운로드 : git clone 주소 폴더이름

  주소는 github 에서 copy

  폴더이름은 선택사항이다 (즉 없어도 됨)

  폴더이름을 줄 경우에는 그 폴더가 새로 생성이 되면서 그 안에 코드들이 다운로드가 되고, 폴더이름을 안 줄 경우 github 프로젝트 이름으로 폴더가 자동으로 생기고 그안에 코드들이 다운로드 된다.

```
PS C:\Users\picbu> cd c:\javaproject
PS C:\javaproject> git clone https://github.com/picbuddy/test.git
Cloning into 'test'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 17 (delta 3), reused 17 (delta 3), pack-reused 0
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (3/3), done.
PS C:\javaproject>
```

fork(my remote repo) → clone(my local repo) → branch(local, option) → commit & push(local) → pull(check from remote) → PR(pull request to owner)

```
PS C:\javaproject> cd test
PS C:\javaproject\test> git branch
* main
PS C:\javaproject\test>
```

1. Github 에서 내 브랜치(branch)만들기

   git checkout -b develop

2. 내 브랜치에 소스코드 업데이트하기

   git add .

   git commit -m "first commit"

   git push origin develop

3. 마스터 브랜치에 소스 가져오기(pull)

   git pull origin develop (branch name)

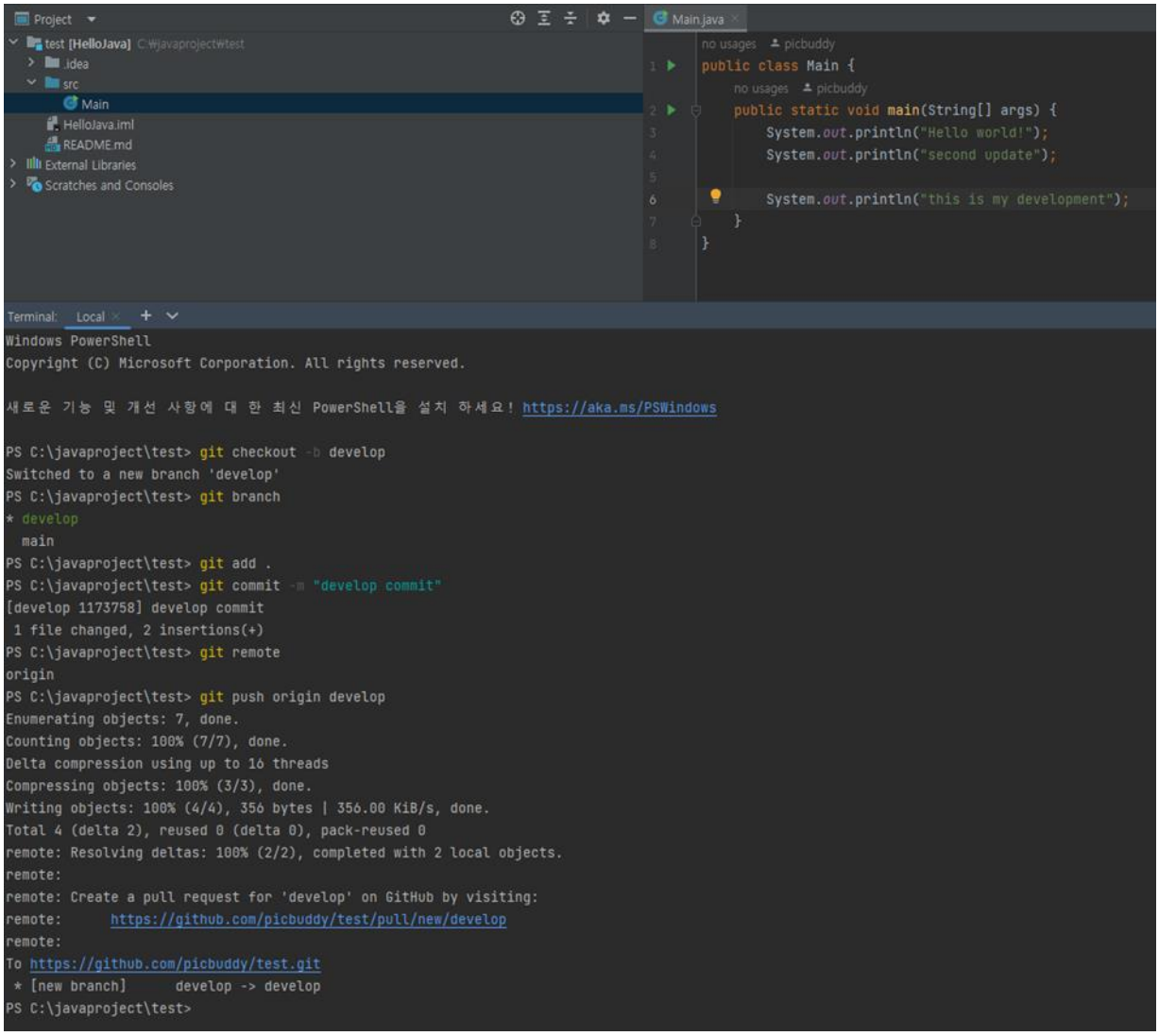
   pull 을 하기전에는 기존에 소스코드들을 commit 을 먼저 해야 한다

4. 브랜치끼리 이동하는 법

   git checkout develop (branch name)

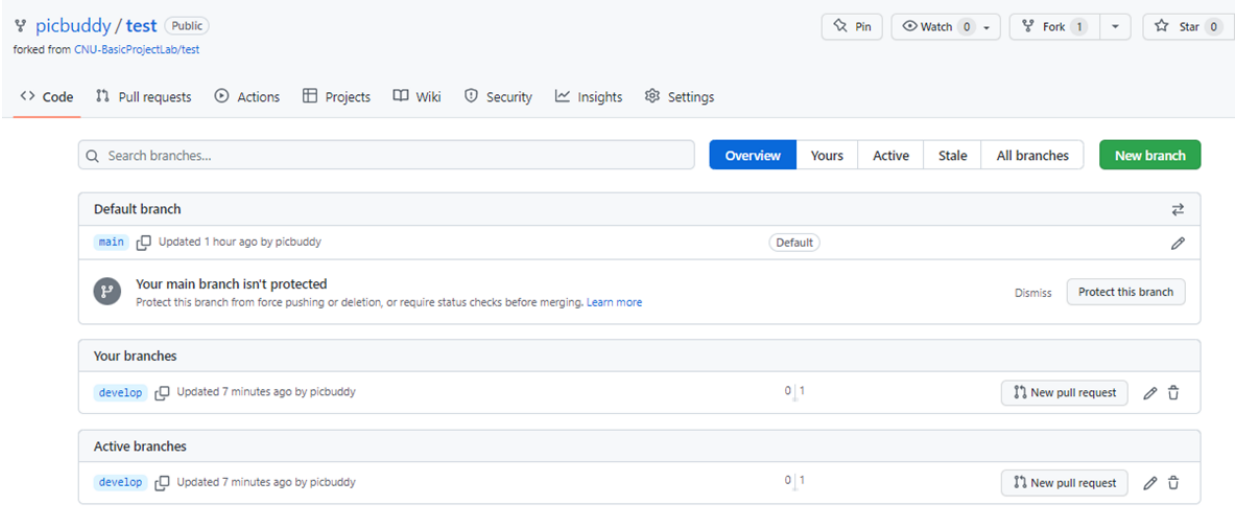
   내 브랜치에서 마스터 브렌치로 이동을 하고 싶거나

   다른 브랜치로 이동하고 싶으면 해당 명령어를 쓰면 된다

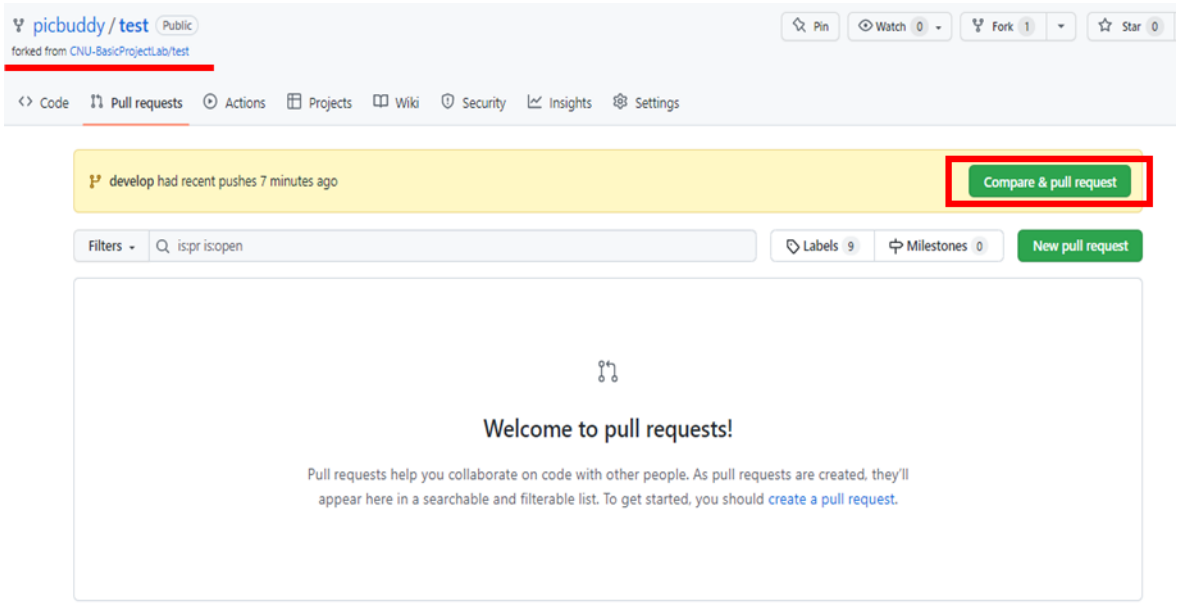

Project
test [HelloJava] C:₩javaproject₩test
.idea
src
Main
HelloJava.iml
README.md
External Libraries
Scratches and Consoles
Main.java
no usages picbuddy
public class Main {
no usages picbuddy
public static void main(String[] args) {
System.out.println("Hello world!");
System.out.println("second update");
System.out.println("this is my development");
}
}
Terminal: Local
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows
PS C:\javaproject\test> git checkout -b develop
Switched to a new branch 'develop'
PS C:\javaproject\test> git branch
* develop
main
PS C:\javaproject\test> git add .
PS C:\javaproject\test> git commit -m "develop commit"
[develop 1173758] develop commit
1 file changed, 2 insertions(+)
PS C:\javaproject\test> git remote
origin
PS C:\javaproject\test> git push origin develop
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 356 bytes | 356.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote: https://github.com/picbuddy/test/pull/new/develop
remote:
To https://github.com/picbuddy/test.git
* [new branch] develop -> develop
PS C:\javaproject\test>

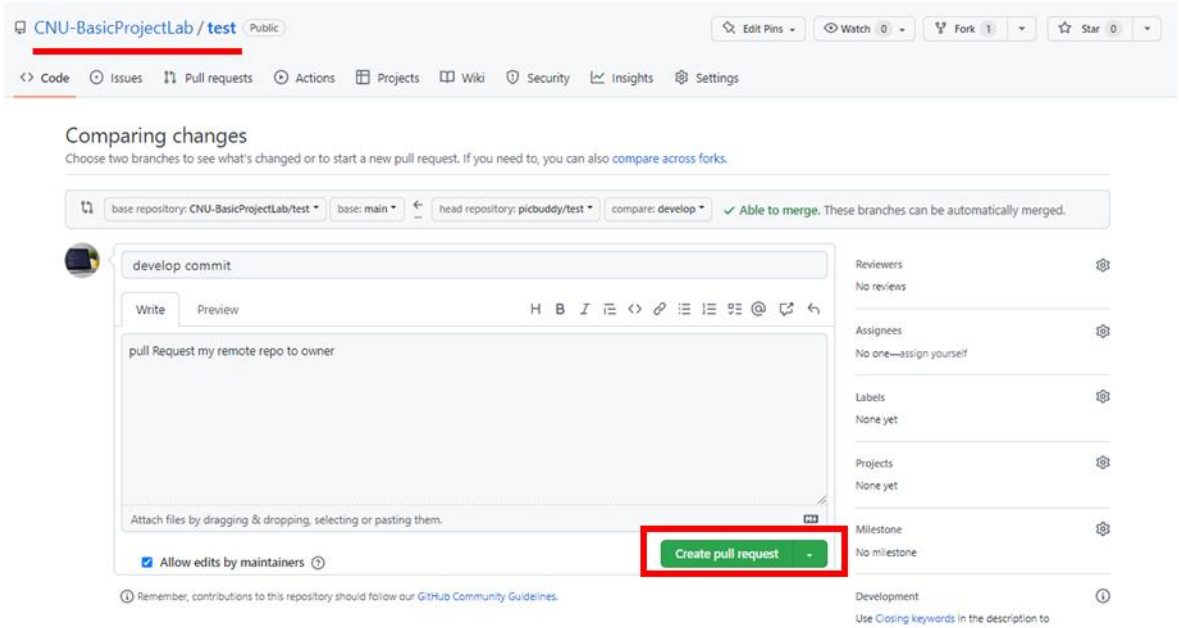
fork(my remote repo) → clone(my local repo) → branch(local, option) → commit&push(local) → pull(check from remote) → PR(pull request to owner)

```
PS C:\javaproject\test> git pull origin main
From https://github.com/picbuddy/test
 * branch            main       -> FETCH_HEAD
Already up to date.
PS C:\javaproject\test> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\javaproject\test> git branch
  develop
* main
PS C:\javaproject\test>
```

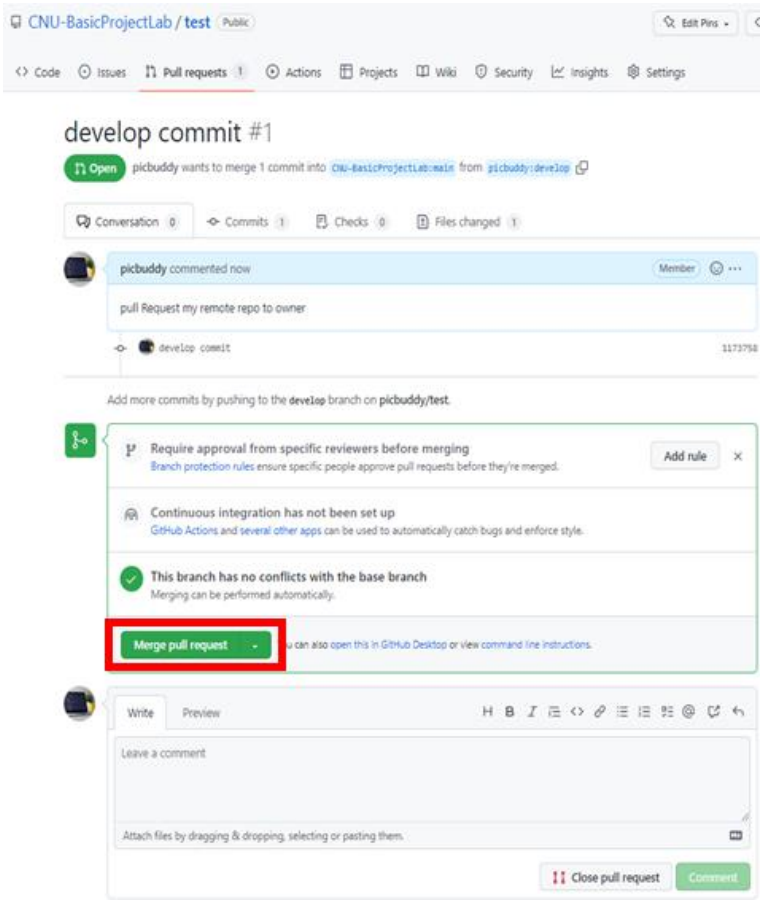


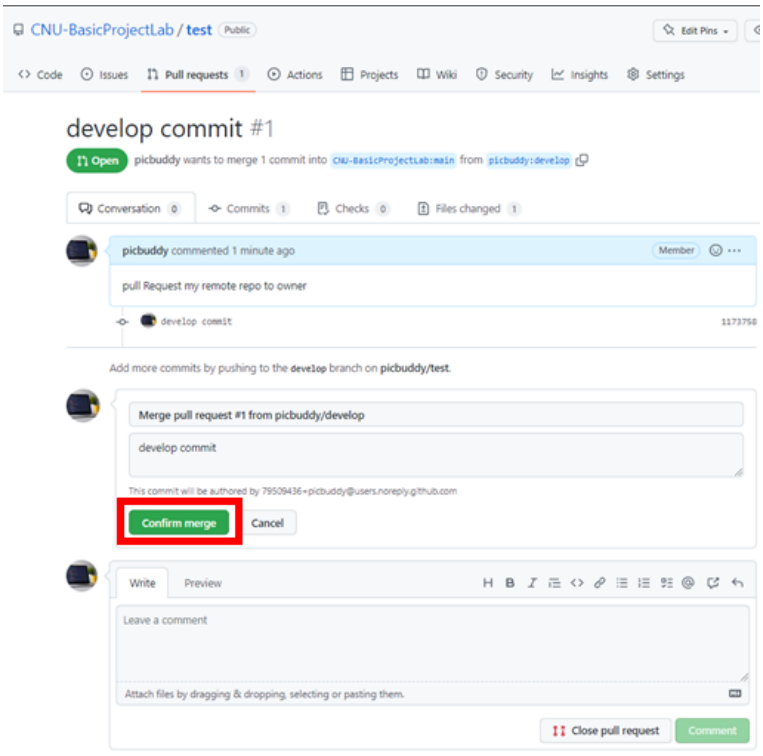
fork(my remote repo) → clone(my local repo) → branch(local,option) → commit & push(local) → pull(check from remote) → PR(pull request to owner)

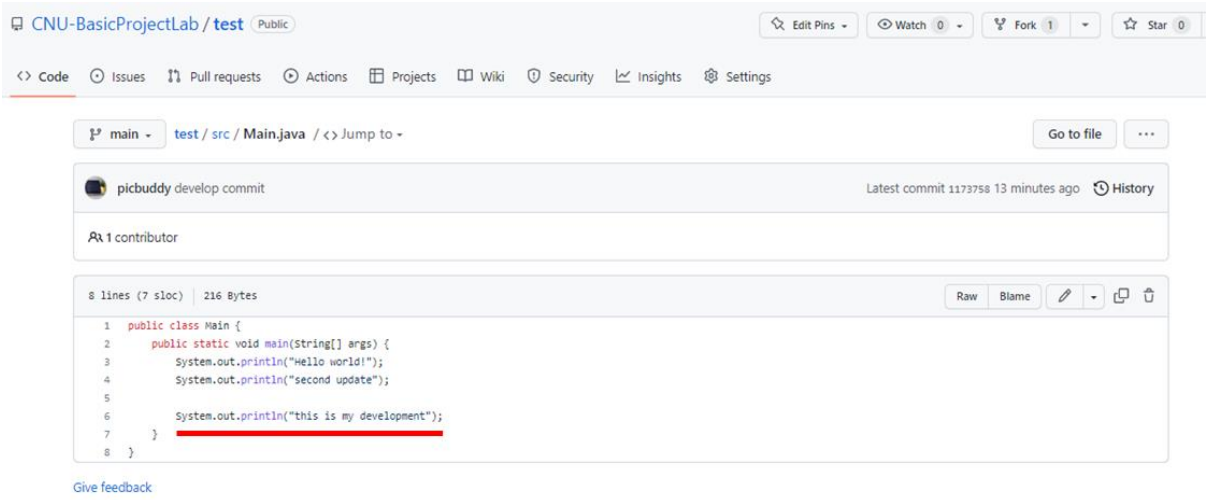




fork(my remote repo) → clone(my local repo) → branch(local, option) → commit & push(local) → pull(check from remote) → PR(pull request to owner) → Merge!!



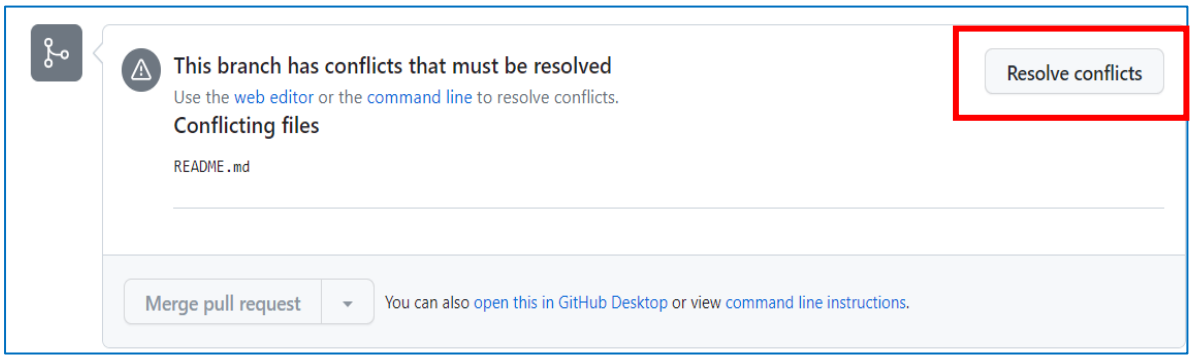


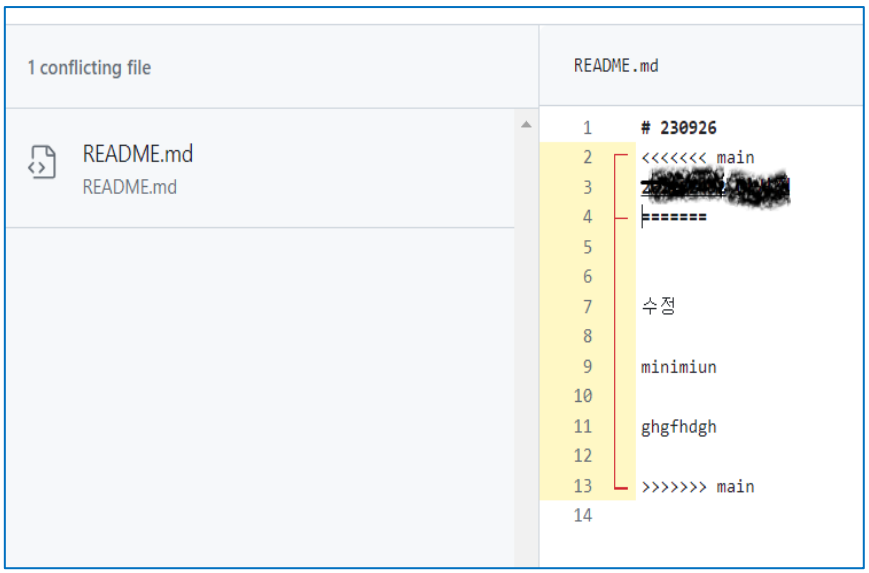


브렌치 삭제

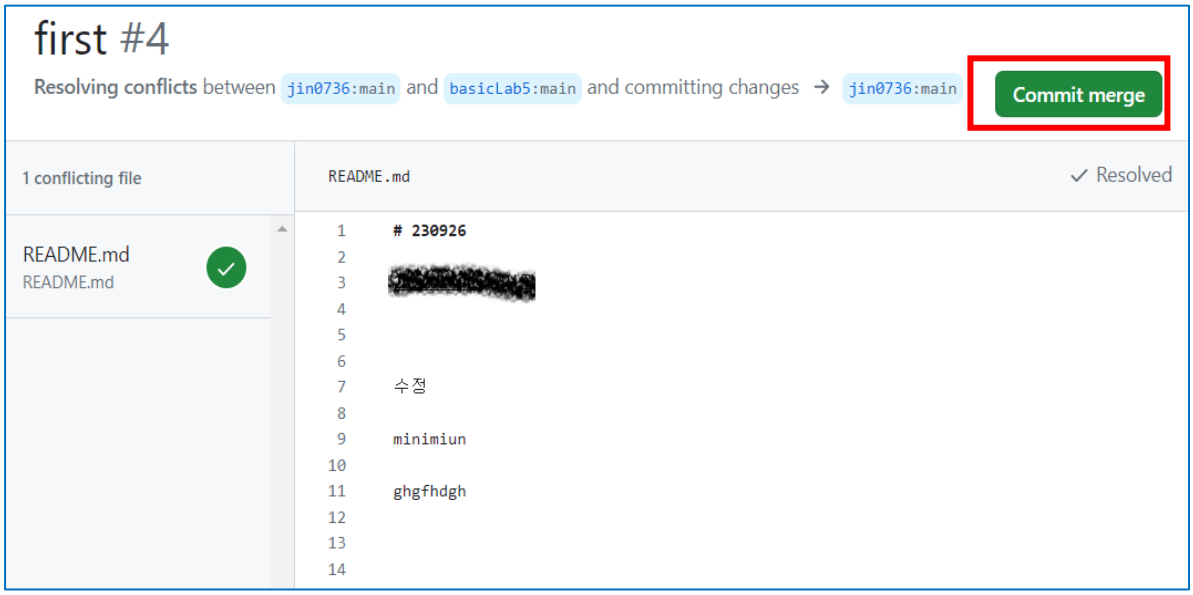
git branch -d <로컬 브랜치 이름>

Resolve conflict






1 conflicting file
README....
README.md
README.md
1 conflict
Prev
Next
Mark as resolved
1 # 230926
2
3
4
5
6
7 수정
8
9 minimiun
10
11 ghgfhdgh
12
13
14


first #4
Resolving conflicts between jin0736:main and basicLab5:main and committing changes → jin0736:main
Commit merge
1 conflicting file
README.md
README.md
README.md
✓ Resolved
1 # 230926
2
3
4
5
6
7 수정
8
9 minimiun
10
11 ghgfhdgh
12
13
14

fork(my remote repo) → clone(my local repo) → branch(local, option) → commit & push(local) → pull(check from remote) → PR(pull request to owner) → Merge!! → fetch (get other's update)

1. **원본 저장소 등록(fork 를 한 원본 저장소)**
   - **등록된 저장소 확인**

     $ git remote -v

     origin https://github.com/~ (fetch)

     origin https://github.com/~ (push)
   - **원격 저장소 upstream 등록**

     $ git remote add upstream https://github.com/~
   - **원본 저장소 upstream 등록 확인**

     $ git remote -v

     origin https://github.com/~ (fetch)

     origin https://github.com/~ (push)

     upstream https://github.com/~ (fetch)

     upstream https://github.com/~ (push)

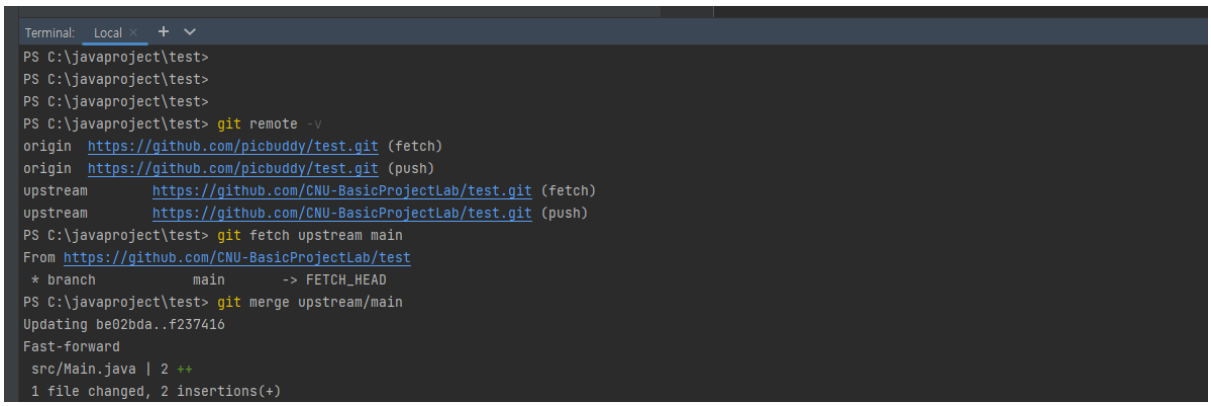
     $ git remote remove origin (삭제 시 참고)
2. **원본 저장소 변경내용 로컬로 가져오기**

   $ git fetch upstream main
3. **원본 저장소와 포크한 저장소 병합**

   $ git merge upstream/main

   **** pull = fetch + merge**
4. **포크한 저장소를 Git 서버에 적용**

   $ git push origin main

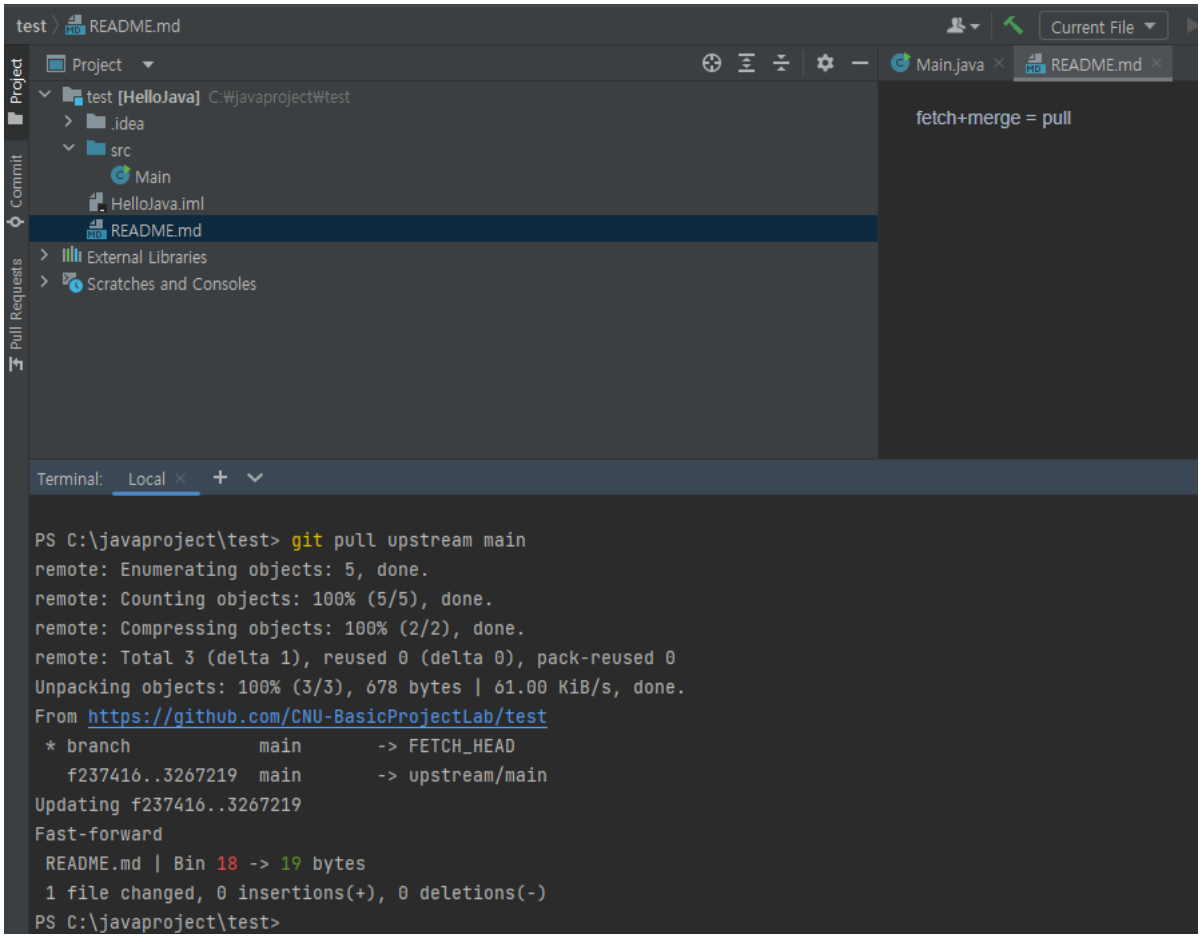


fork(my remote repo) → clone(my local repo) → branch(local, option) →

commit&push(local) → pull(check from remote)→ PR(pull request to owner) →

Merge!! → fetch+merge = pull (get other's update)

- fetch 는 원격 저장소에 변경사항이 있는지 확인만 하고, 변경된 데이터를 로컬 Git 에 실제로 가져오지는 않습니다.
- 반면 git pull 은 원격 저장소에서 변경된 메타데이터 정보를 확인할 뿐만 아니라 최신 데이터를 복사하여 로컬 Git 에 가져옵니다.



[실습]

https://github.com/CNU-BasicProjectLab/test 에 PR 보내기

4. Dart

4-1. Dart?!

다트 개요

Dart는 모든 플랫폼에서 빠른 앱을 개발하기 위한 클라이언트 최적화 언어입니다. 목표는 앱 프레임워크를 위한 유연한 실행 런타임 플랫폼 과 함께 다중 플랫폼 개발을 위한 가장 생산적인 프로그래밍 언어를 제공하는 것입니다 .

언어는 기술 범위, 즉 언어의 기능과 강점을 형성하는 개발 중에 이루어진 선택 으로 정의됩니다 . Dart는 다양한 컴파일 대상(웹, 모바일 및 데스크톱)에서 개발(1초 미만의 상태 저장 핫 리로드) 및 고품질 프로덕션 경험을 모두 우선시하여 클라이언트 개발에 특히 적합한 기술 범위를 위해 설계되었습니다.

Dart

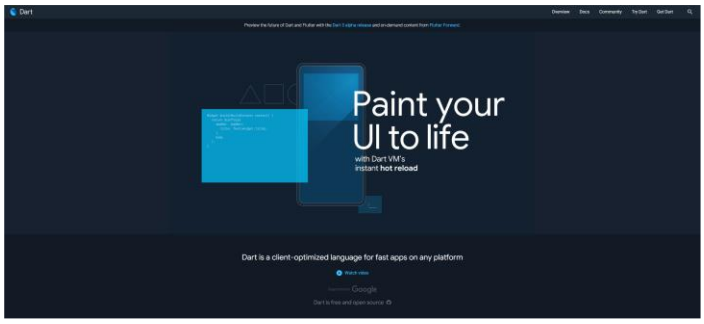
Dart는 또한 Flutter 의 기반을 형성합니다 . Dart는 Flutter 앱을 지원하는 언어와 런타임을 제공하지만 Dart는 코드 서식 지정, 분석 및 테스트와 같은 많은 핵심 개발자 작업도 지원합니다.

다트: 플랫폼

Dart의 컴파일러 기술을 사용하면 다양한 방식으로 코드를 실행할 수 있습니다.

- **기본 플랫폼** : 모바일 및 데스크톱 장치를 대상으로 하는 앱의 경우 Dart에는 JIT(Just-In-Time) 컴파일 기능이 있는 Dart VM 과 머신 코드 생성을 위한 AOT(Ahead-of-Time) 컴파일러가 모두 포함되어 있습니다.
- **웹 플랫폼** : 웹을 대상으로 하는 앱의 경우 Dart는 개발 또는 프로덕션 목적으로 컴파일할 수 있습니다. 웹 컴파일러는 Dart 를 JavaScript로 변환합니다.

https://dart.dev/



Dart

패러다임	객체 지향 프로그래밍
개발자	구글
발표일	2011년 10월 10일(11년 전)[1]
최근 버전	2.19.2[2]
최근 버전 출시일	2023년 2월 8일(7일 전)
미리보기 버전 출시일	2021년 12월 14일(14개월 전)[3]
자료형 체계	선택적
웹사이트	https://dart.dev
영향을 받은 언어	
자바스크립트, 자바	

JIT 컴파일

문서 토론

위키백과, 우리 모두의 백과사전.

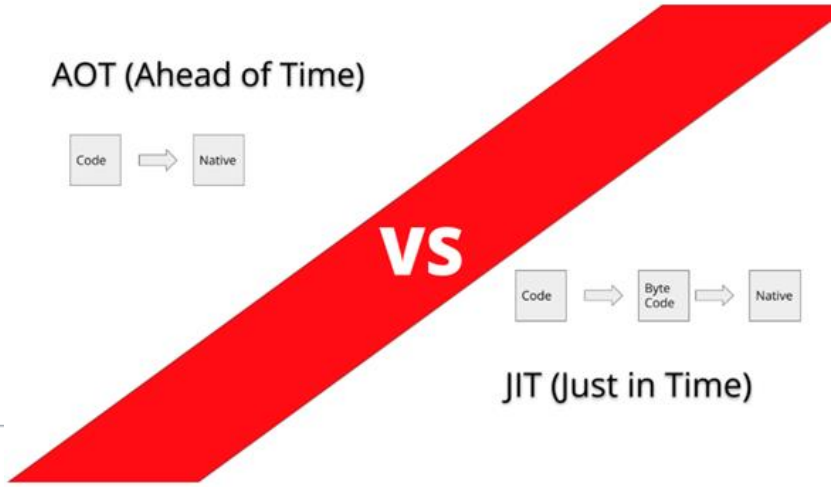
JIT 컴파일(just-in-time compilation) 또는 **동적 번역**(dynamic translation)은 프로그램을 실제 실행하는 시점에 기계어로 번역하는 컴파일 기법이다.

전통적인 입장에서 컴퓨터 프로그램을 만드는 방법은 두 가지가 있는데, 인터프리트 방식과 정적 컴파일 방식으로 나눌 수 있다. 이 중 인터프리트 방식은 실행 중 프로그래밍 언어를 읽어가면서 해당 기능에 대응하는 기계어 코드를 실행하며, 반면 정적 컴파일은 실행하기 전에 프로그램 코드를 기계어로 번역한다.

JIT 컴파일러는 두 가지의 방식을 혼합한 방식으로 생각할 수 있는데, 실행 시점에서 인터프리트 방식으로 기계어 코드를 생성하면서 그 코드를 캐싱하여, 같은 함수가 여러 번 불릴 때 매번 기계어 코드를 생성하는 것을 방지한다.

최근의 자바 가상 머신과 .NET, V8(node.js)에서는 JIT 컴파일을 지원한다. 즉, 자바 컴파일러가 자바 프로그램 코드를 바이트코드로 변환한 다음, 실제 바이트코드를 실행하는 시점에서 자바 가상 머신이 바이트코드를 JIT 컴파일을 통해 기계어로 변환한다.

https://ko.wikipedia.org/wiki/JIT_%EC%BB%B4%ED%8C%8C%EC%9D%BC#%EA%B0%81%EC%A3%BC



AOT 컴파일

문서 토론

위키백과, 우리 모두의 백과사전.

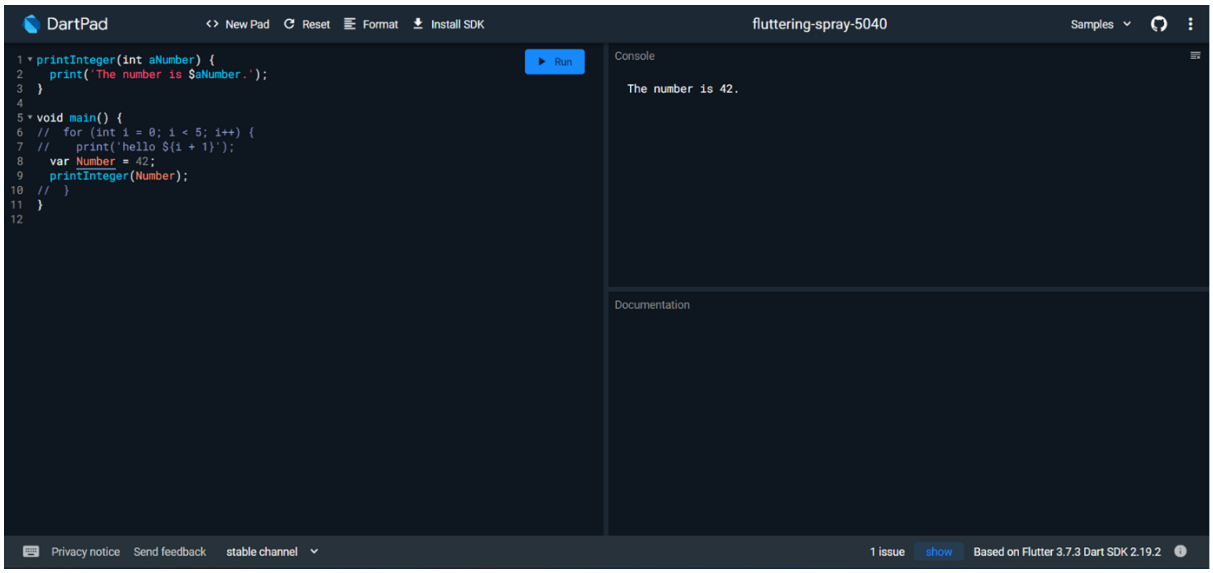
AOT 컴파일(ahead-of-time compile)은 목표 시스템의 기계어와 무관하게 중간 언어 형태로 배포된 후 목표 시스템에서 인터프리터나 JIT 컴파일 등 기계어 번역을 통해 실행되는 중간 언어를 미리 목표 시스템에 맞는 기계어로 번역하는 방식을 지칭한다. 이런 중간 언어로는 자바 바이트코드, 공통 중간 언어(Common Intermediate Language), IBM System/38 혹은 IBM System i의 기술 독립적 머신 인터페이스(Technology Independent Machine Interface)가 있으며 학계에서도 마이클 프란즈(Michael Franz)[1]가 제안해 오베론 시스템의 일부 구현에서 사용된 슬림 바이너리(Slim Binaries)와 같은 제안이 있었다.

일반적으로 중간 언어를 사용하는 시스템은 실행 시간에 JIT 컴파일과 같은 기법을 통해 실행 시에만 얻을 수 있는 프로그램 분석 정보를 사용하여 높은 성능을 달성할 수 있다. 특히 객체 지향 언어나 스크립트 언어 같이 동적인 언어인 경우 효과적이다. 하지만 실행 시 프로그램 분석과 컴파일을 함께 수행하는데 추가 메모리 및 CPU 사이클이 필요한 단점이 있다. 따라서 AOT 컴파일은 이에 대한 보완책으로 사용되고 있다.

https://ko.wikipedia.org/wiki/AOT_%EC%BB%B4%ED%8C%8C%EC%9D%BC

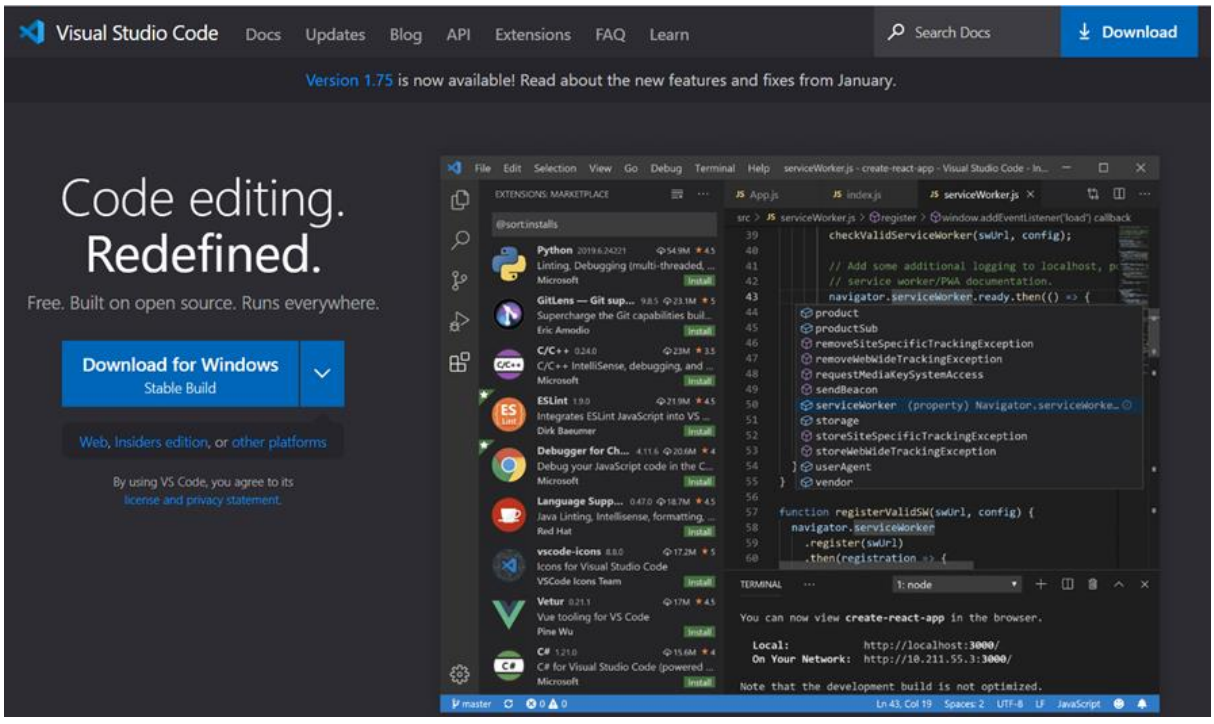
4-2. Dart 언어의 기본 문법 학습

DartPad (https://dartpad.dev/?)



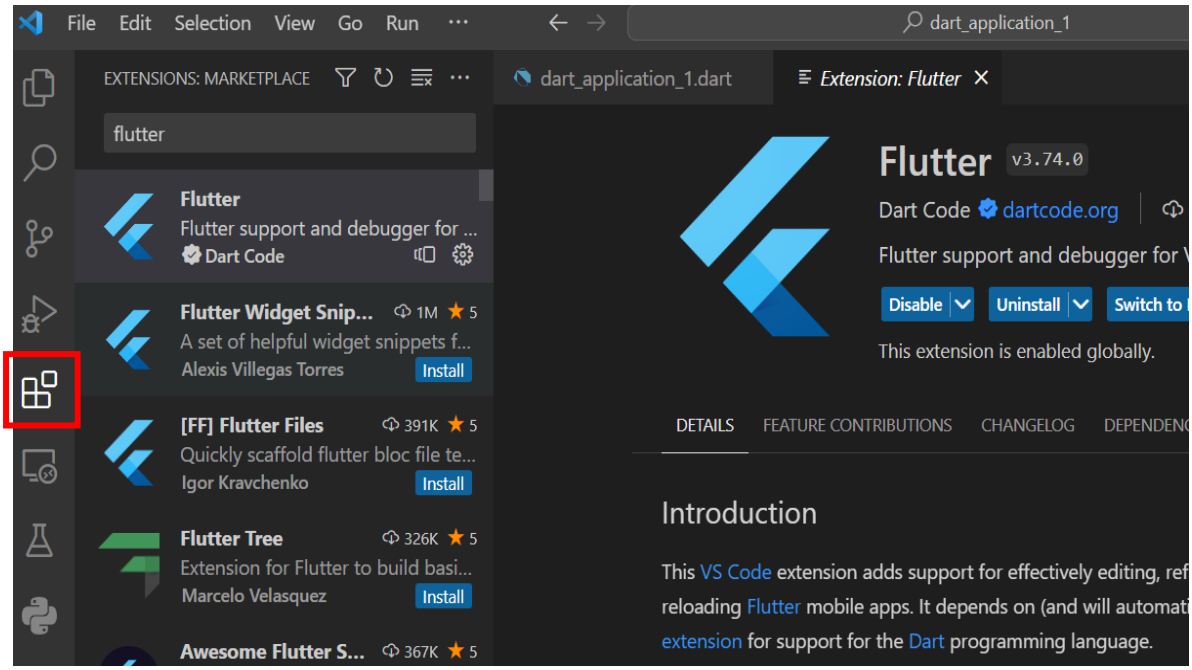
4-3. 예제(VS CODE + Dart)

https://code.visualstudio.com/



Flutter install (https://flutter.dev/)

Flutter SDK 를 설치하면 Dart SDK 가 자동 설치되므로 편함 (추후 VS CODE 에서 dart-sdk 경로 지정해야 함)



APPENDIX : VS CODE extension 에서 Dart 설치할 경우 dart sdk (https://dart.dev/get-dart)

도 설치해야 하는데 chocolatey 로 설치해야 해서 번거롭다 (링크 참조)

- Chocolatey 로 설치하는 방법

https://devshin93.tistory.com/m/142

- sdk zip 파일로 설치하는 방법

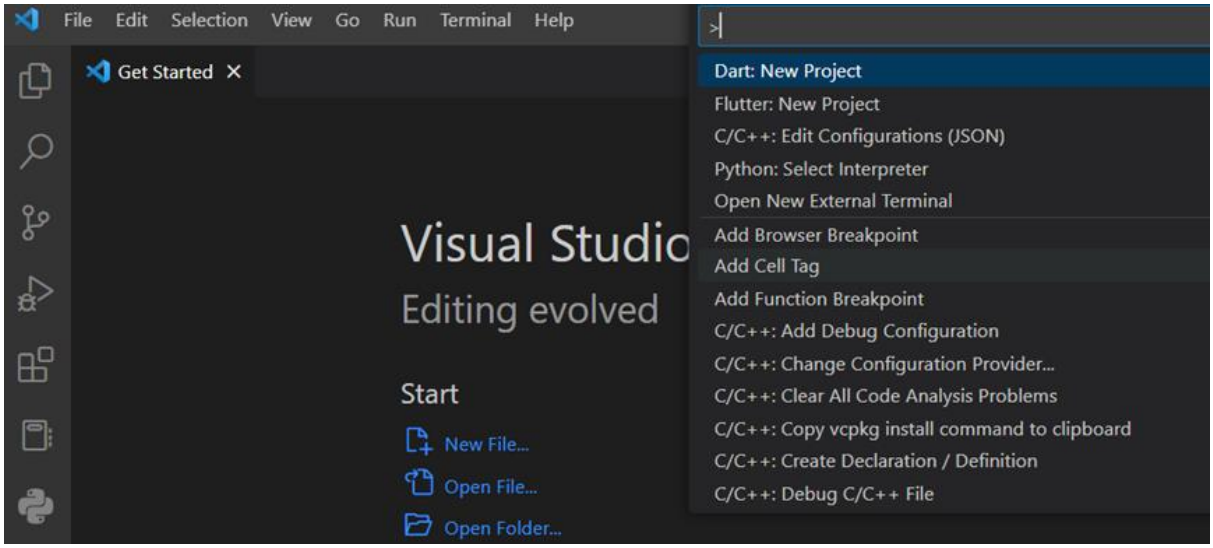
https://www.geeksforgeeks.org/dart-installation-and-setup-in-visual-studio-code/amp/

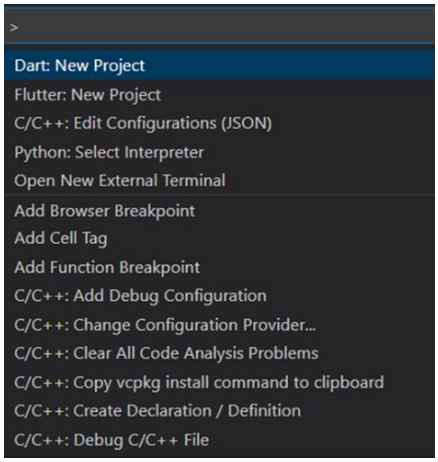
- MAC 에서 VSCODE, Flutter(with dart) 설치하는 방법

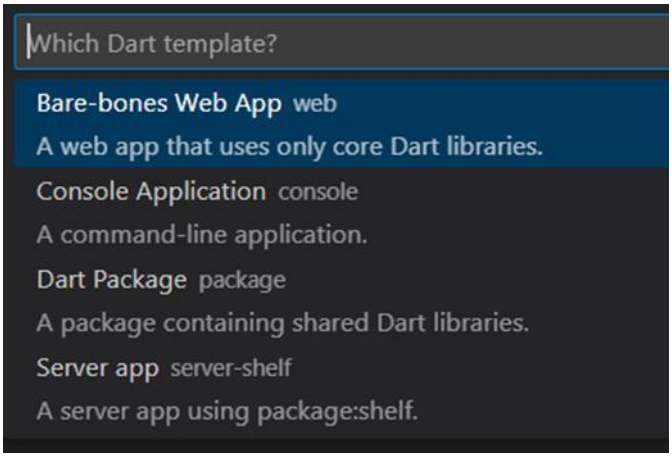
https://proimaginer.tistory.com/m/64

VSCODE 에서 Dart 프로젝트 만들기

ctrl+shift+p → Dart:New Project → Console Application → 폴더 선택 → Project 이름

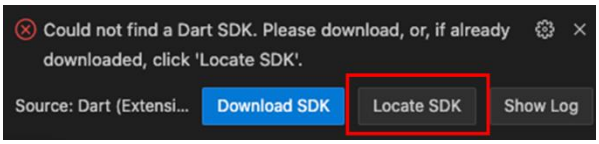


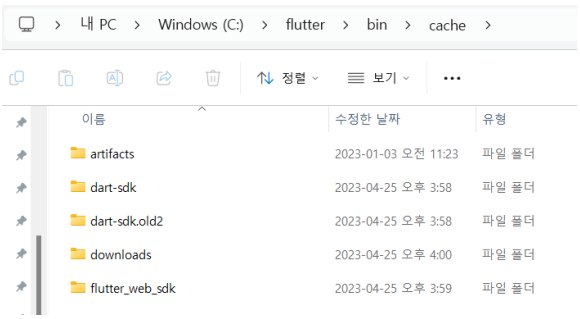




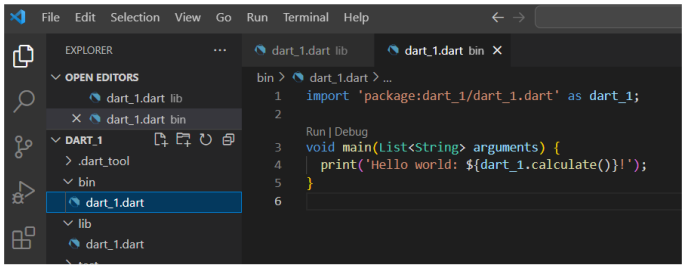
Flutter SDK 폴더에 dart-sdk 경로 지정!

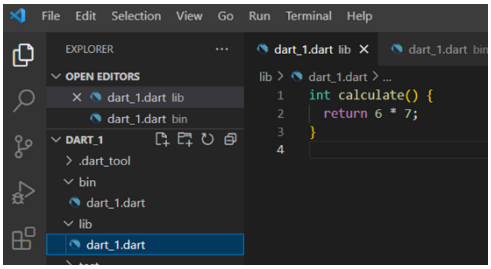
C:₩flutter₩bin₩cache₩dart-sdk



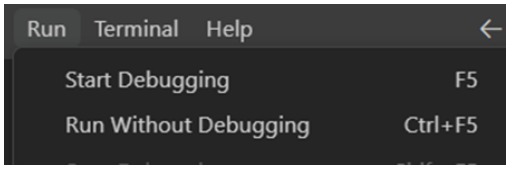


Default Project, sample code #1



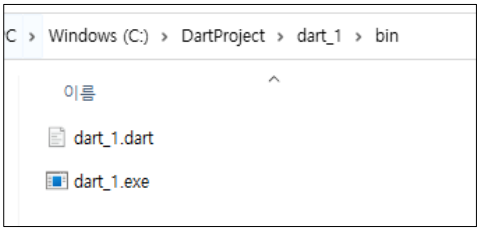


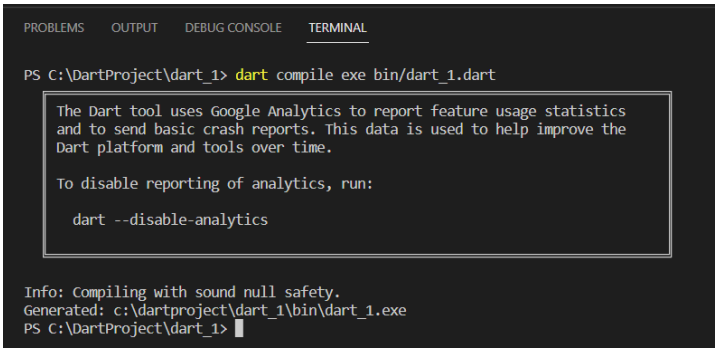
Run



Compile

https://dart.dev/tools/dart-compile





하위 명령	산출	추가 정보
exe	자체 포함된 실행 파일	머신 코드로 컴파일된 소스 코드와 작은 Dart 런타임을 포함하는 독립 실행형 아키텍처별 실행 파일입니다 . *더 알아보기.*
aot-snapshot	AOT 모듈	**기계 코드로 컴파일된 소스 코드를 포함하지만 Dart 런타임은 포함하지 않는** 아키텍처별 파일입니다 . *더 알아보기.*
jit-snapshot	JIT 모듈	모든 소스 코드의 중간 표현과 프로그램의 교육 실행 중에 실행된 소스 코드의 최적화된 표현이 포함된 아키텍처별 파일입니다. JIT 컴파일된 코드는 훈련 데이터가 양호하면 AOT 코드보다 더 빠른 최고 성능을 가질 수 있습니다. *더 알아보기.*
kernel	커널 모듈	소스 코드의 이식 가능한 중간 표현입니다 . *더 알아보기.*
js	자바스크립트	소스 코드에서 컴파일된 배포 가능한 JavaScript 파일입니다. *더 알아보기.*

exe다음은 자체 포함된 실행 파일(myapp.exe) 을 생성하기 위해 하위 명령 을 사용하는 예입니다 .

```
$ dart compile exe bin/myapp.dart
Generated: /Users/me/myapp/bin/myapp.exe
```

다음 예에서는 aot-snapshot하위 명령을 사용하여 AOT(ahead-of-time) 컴파일 모듈( )을 생성합니다 myapp.aot. 그런 다음 dartaotruntime명령 (Dart 런타임 제공)을 사용하여 AOT 모듈을 실행합니다.

```
$ dart compile aot-snapshot bin/myapp.dart
Generated: /Users/me/myapp/bin/myapp.aot
$ dartaotruntime bin/myapp.aot
```

출력 파일의 경로를 지정하려면 -oor --output옵션을 사용하십시오.

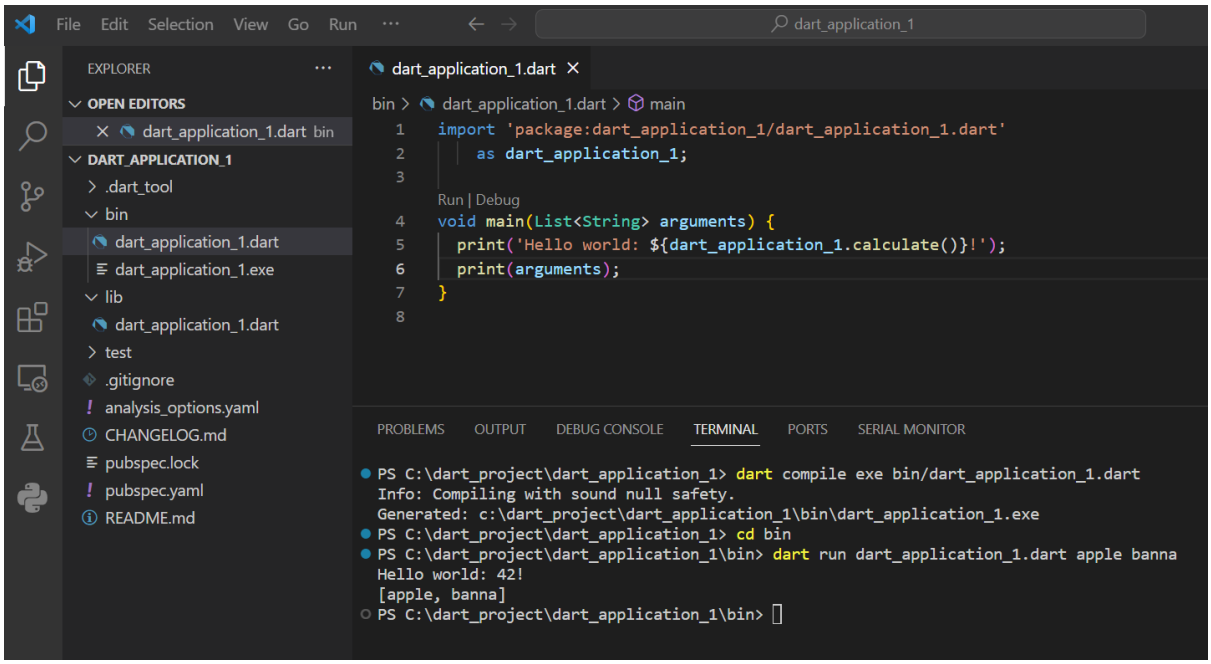
```
$ dart compile exe bin/myapp.dart -o bin/runme
```

추가 옵션 및 사용 정보를 보려면 다음을 실행하십시오 dart compile [<subcommand>] --help.

```
$ dart compile exe --help
```

이 dart compile명령은 dart2native, dart2aot및 dart2js명령을 대체합니다.

self-contained executable file



```
PS C:\dart_project\dart_ex\dart_1\bin> ./dart_1.exe apple banna
Hello world: 42!
[apple, banna]
PS C:\dart_project\dart_ex\dart_1\bin>
```

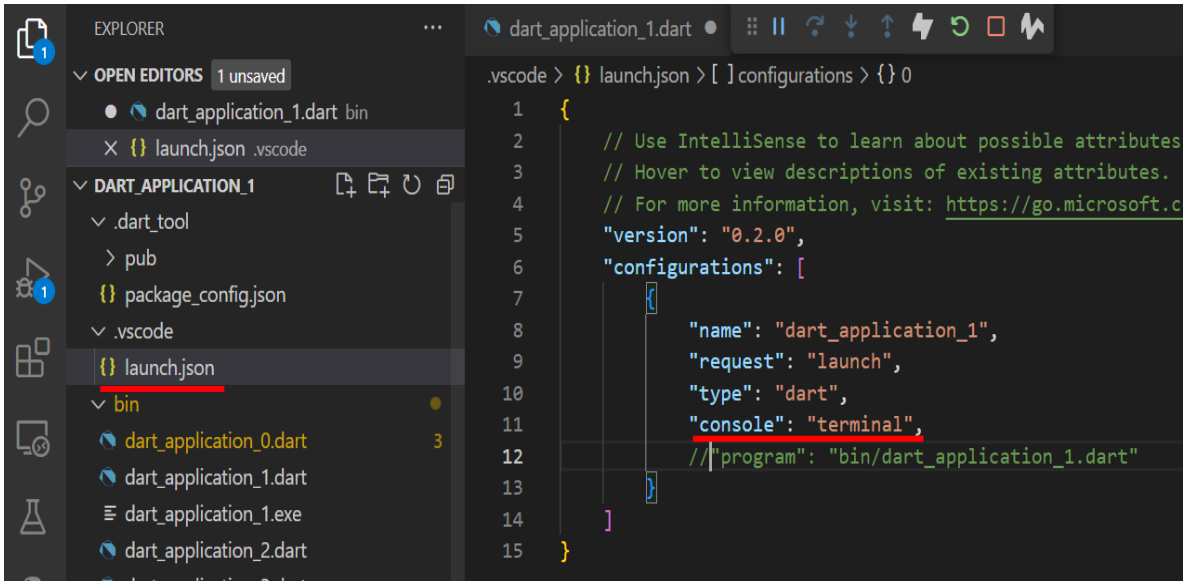
실행파일 더블 클릭하여 실행 후 바로 close 되는 것 방지하기

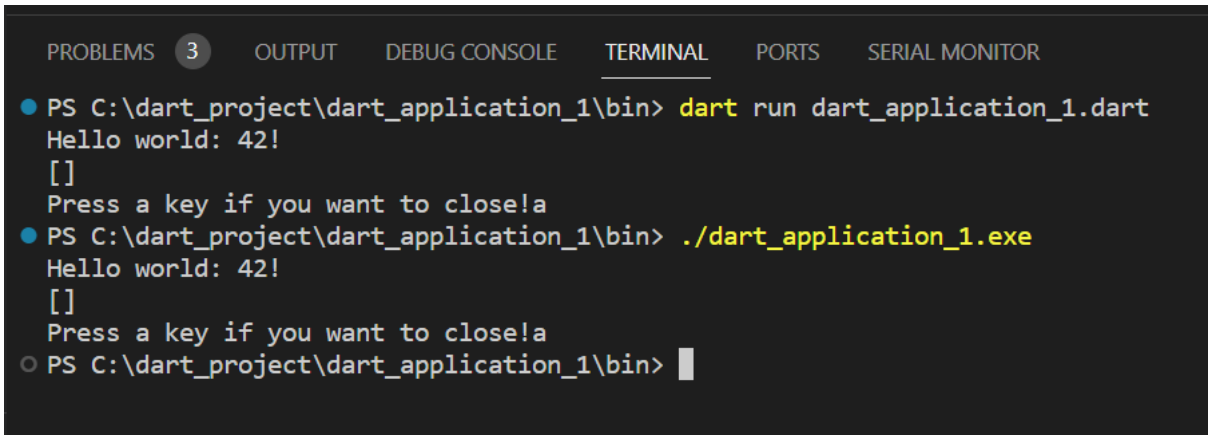
```
bin > dart_application_1.dart > main
1   import 'dart:io';
2
3   import 'package:dart_application_1/dart_application_1.dart'
4       as dart_application_1;
5
    Run | Debug
6   void main(List<String> arguments) async {
7     print('Hello world: ${dart_application_1.calculate()}!');
8     print(arguments);
9
10    stdout.write("Press a key if you want to close!");
11    stdin.readLineSync();
12  }
13
```

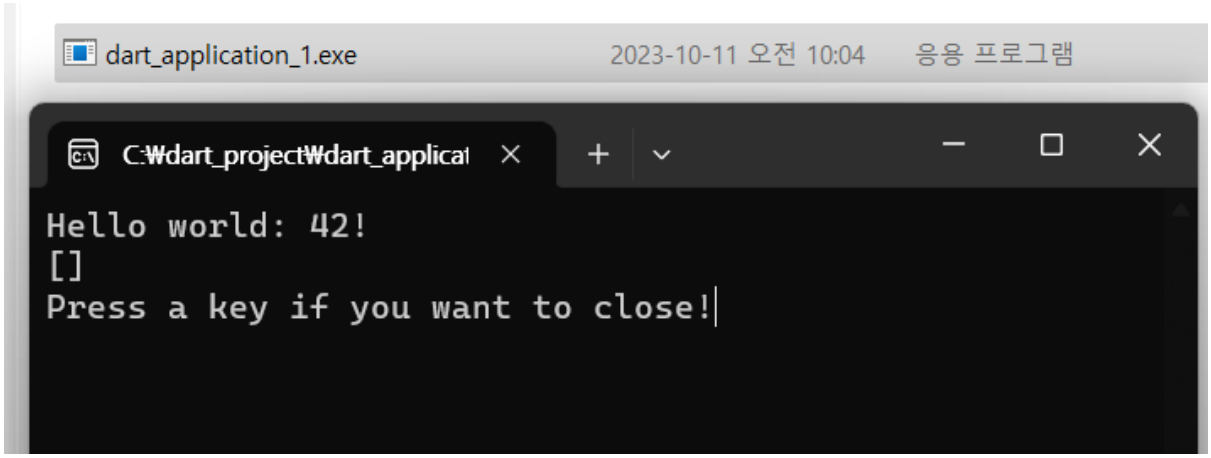
dart:io 는 VSCODE **DebugConsole** 에서 실행 시 error 발생 → Terminal 에서 실행해야 함.

(launch.json 수정 필요)

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SERIAL MONITOR
Connecting to VM Service at ws://127.0.0.1:52507/rSg-ReZzabI=/ws
Hello world: 42!
[]
Press a key if you want to close!
a
UnimplementedError: Global evaluation not currently supported
```

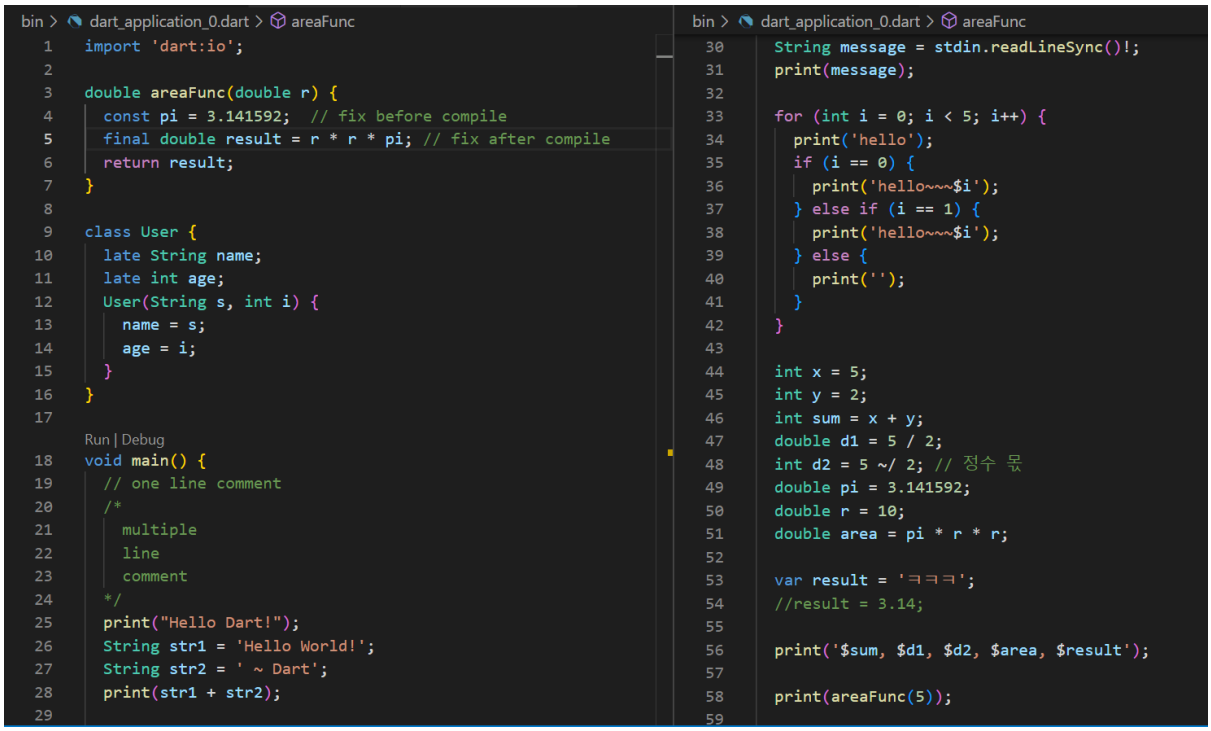






https://dart.dev/language

Language Tour : 입출력, 변수, 반복, 조건, 함수, 클래스



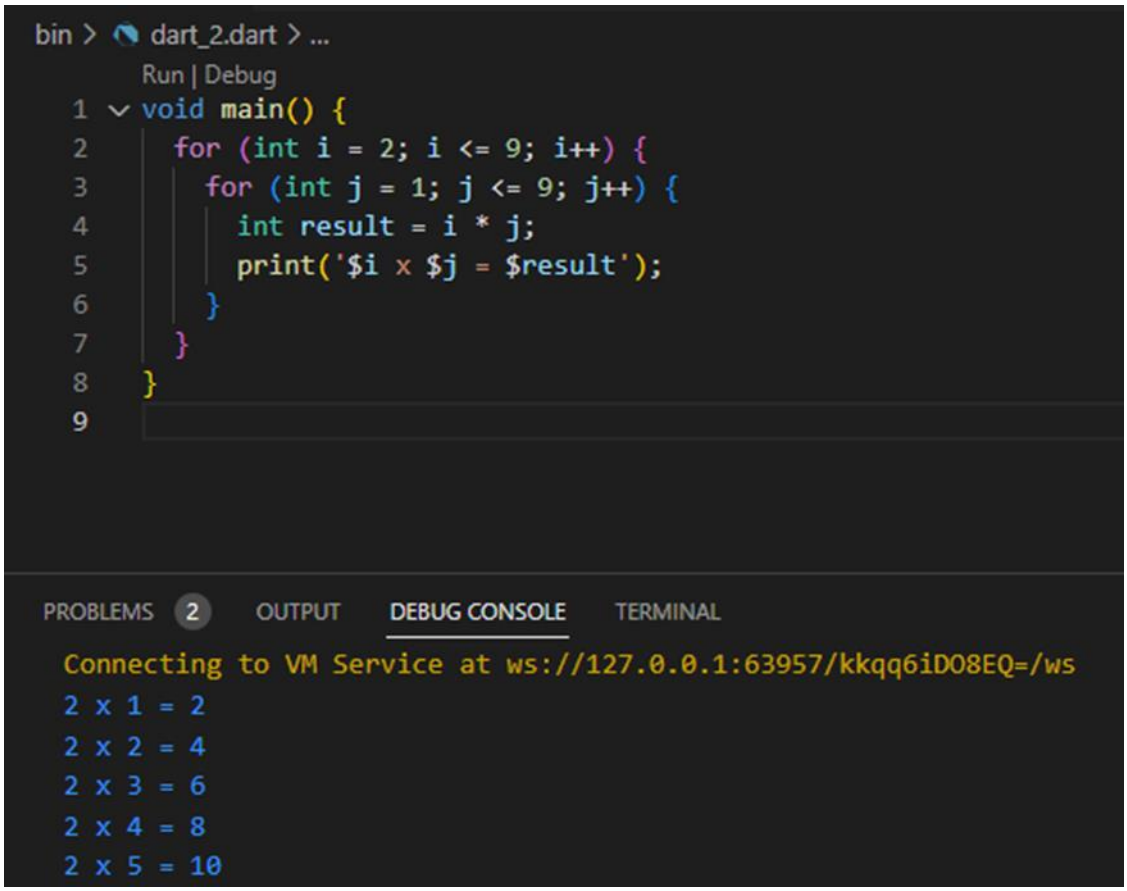
```dart
import 'dart:io';

double areaFunc(double r) {
  const pi = 3.141592;  // fix before compile
  final double result = r * r * pi; // fix after compile
  return result;
}

class User {
  late String name;
  late int age;
  User(String s, int i) {
    name = s;
    age = i;
  }
}

Run | Debug
void main() {
  // one line comment
  /*
    multiple
    line
    comment
  */
  print("Hello Dart!");
  String str1 = 'Hello World!';
  String str2 = ' ~ Dart';
  print(str1 + str2);

  String message = stdin.readLineSync()!;
  print(message);

  for (int i = 0; i < 5; i++) {
    print('hello');
    if (i == 0) {
      print('hello~~~$i');
    } else if (i == 1) {
      print('hello~~~$i');
    } else {
      print('');
    }
  }

  int x = 5;
  int y = 2;
  int sum = x + y;
  double d1 = 5 / 2;
  int d2 = 5 ~/ 2; // 정수 몫
  double pi = 3.141592;
  double r = 10;
  double area = pi * r * r;

  var result = 'ㅋㅋㅋ';
  //result = 3.14;

  print('$sum, $d1, $d2, $area, $result');

  print(areaFunc(5));
```

```dart
  // arrays are List objects
  var list = [
    'Car',
    'Boat',
    'Plane',
  ];
  List<int> list2 = [1, 2, 3, 4, 5];

  print(list);
  print(list2);
  print(list[2]);

  // A set in Dart is an unordered collection of unique items
  var halogens = {'apple', 'banana', 'melon', 'mango', 'orange'};
  Set<String> s = {'1', '2', '2', '2', '5'};
  print(halogens);
  print(s);

  // a map is an object that associates keys and values
  var gifts = {
    // Key:    Value
    '1': 'diamond',
    '2': 'sappire',
    '3': 'gold'
  };
  print(gifts);

  Map<int, double> m = {1: 1.0, 2: 2.0, 3: 3.0};
  print(m[1]);
```

```dart
  };
  print(gifts);

  Map<int, double> m = {1: 1.0, 2: 2.0, 3: 3.0};
  print(m[1]);

  User u = User("kim", 30);
  print('${u.name}, ${u.age}');
}
```

Call by value?

구구단 프로그램 만들기, sample code #2



Built-in types

The Dart language has special support for the following:

- Numbers (`int`, `double`)
- Strings (`String`)
- Booleans (`bool`)
- Lists (`List`, also known as *arrays*)
- Sets (`Set`)
- Maps (`Map`)
- Runes (`Runes`; often replaced by the `characters` API)
- Symbols (`Symbol`)
- The value `null` (`Null`)

https://dart.dev/guides/language/language-tour#built-in-types

Dart does not pass by reference. Dart is a pass-by-value language.

Null Safety, sample code #3

https://dart.dev/null-safety

https://dart.dev/null-safety/understanding-null-safety

Null safety 는 2.12.0 버전 이상부터 지원

Null 로 설정된 변수에 대한 의도하지 않은 액세스로 인해 발생하는 오류를 방지합니다

변수가 null 값을 가질 수 있음을 나타내려면 ? 를 추가하십시오

```dart
bin > dart_application_3.dart > ...
Run | Debug
void main() {
  List<String?> list = ["apple", "banna", "melon", null];

  for (int i = 0; i < list.length; i++) {
    print(list[i]);
  }
}
```

```dart
nullex.dart 1
bin > nullex.dart > RecommendSystem
// null-safety 적용 이전
/*
bool isEmpty(String string) => string.length == 0;
void main(){
    isEmpty(null);
}
*/

// null safety 적용 이후
requireNonNullableVariable(int nonNullableVar){
    print(nonNullableVar);
}
Run | Debug
void main(){
    int? nullableVar = null; // null을 대입해도 무방
    requireNonNullableVariable(nullableVar); // 컴파일 에러!!
}

class RecommendSystem{
  final bool isRecommend;
  final String? reasonOfRecommend;

  RecommendSystem.yes()
    : isRecommend = true,
      reasonOfRecommend = "convenience";
  RecommendSystem.no()
    : isRecommend = false,
      reasonOfRecommend = null;

  @override
  String toString(){
    if(isRecommend == false) return "I don't recommend this system";
    return "I recoomend this system because of ${reasonOfRecommend!.toUpperCase()}"; // 사용 가능!
  }
}
```

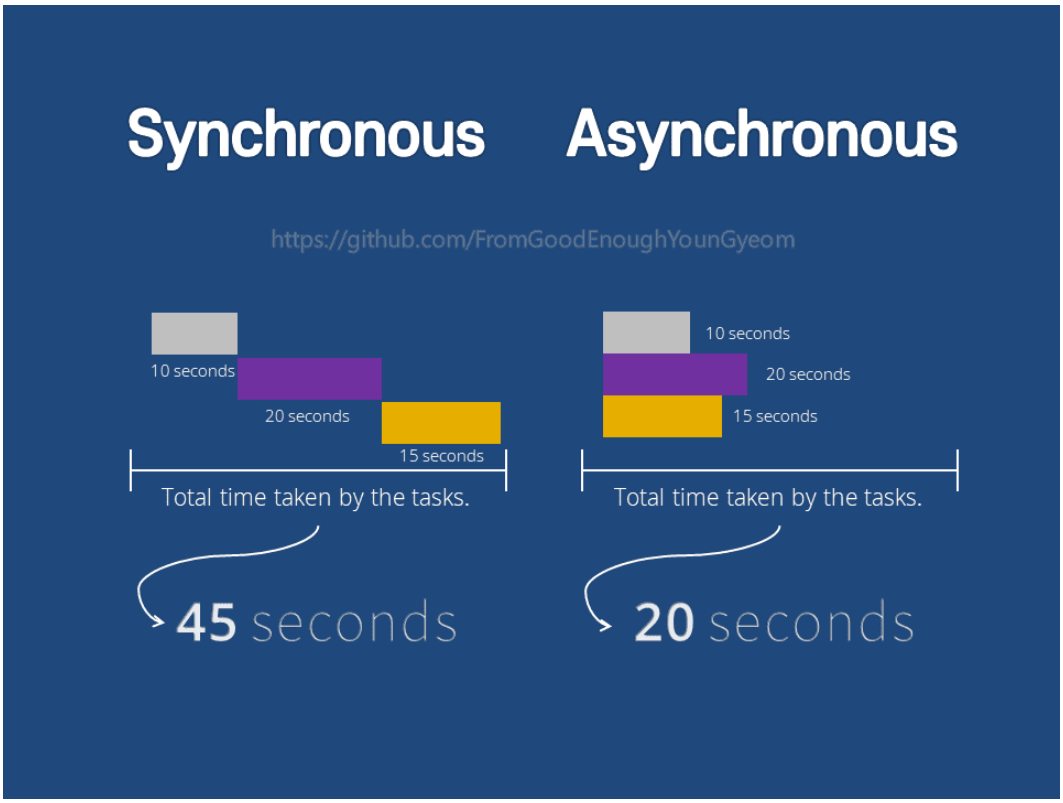
Asynchronous programming: future, async, await sample code #4

다트는 비동기 처리를 지원하는 언어입니다.

비동기(asynchronous)란 언제 끝날지 모르는 작업(네트워크에서 데이터를 가져오거나 데이터베이스 쓰기, 파일 읽기등)을 기다리지 않고 다음 작업을 처리하게 하는 것을 의미합니다

https://dart.dev/codelabs/async-await

https://goodenoughyoungyeom.netlify.app/Web%20Development/[Web%20Development]%EB%8F%99%EA%B8%B0%EC%99%80%EB%B9%84%EB%8F%99%EA%B8%B0%EC%B2%98%EB%A6%AC/



Future<T> ????

```
bin > dart_application_4.dart > ...
1   Future<void> fetchUserOrder() {
2     // Imagine that this function is fetching user info from another service or database.
3     return Future.delayed(const Duration(seconds: 2), () => print('Large Latte'));
4   }
5
    Run | Debug
6   void main() {
7     fetchUserOrder();
8     print('Fetching user order...');
9   }
10
```

Sync program → 4542 millisconds

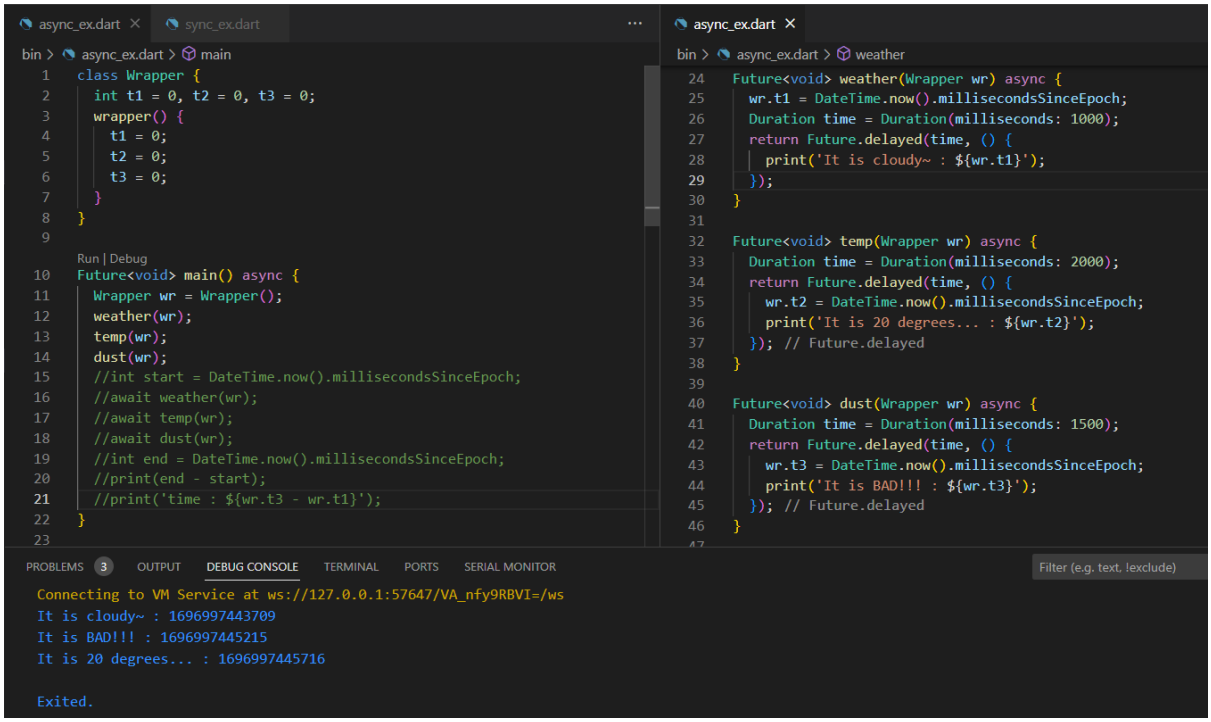
bin > sync_ex.dart > main

```dart
import 'dart:io';

void weather() {
  Duration t = Duration(milliseconds: 1000);
  sleep(t);
  print('It is cloudy.');
}

void temp() {
  Duration t = Duration(milliseconds: 2000);
  sleep(t);
  print('It is 20 degrees.');
}

void dust() {
  Duration t = Duration(milliseconds: 1500);
  sleep(t);
  print('It is bad.');
}

Run | Debug
void main() {
  int start = DateTime.now().millisecondsSinceEpoch;
  weather();
  temp();
  dust();
  int end = DateTime.now().millisecondsSinceEpoch;
  print(end - start);
}
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS ··· Filter (e.g.

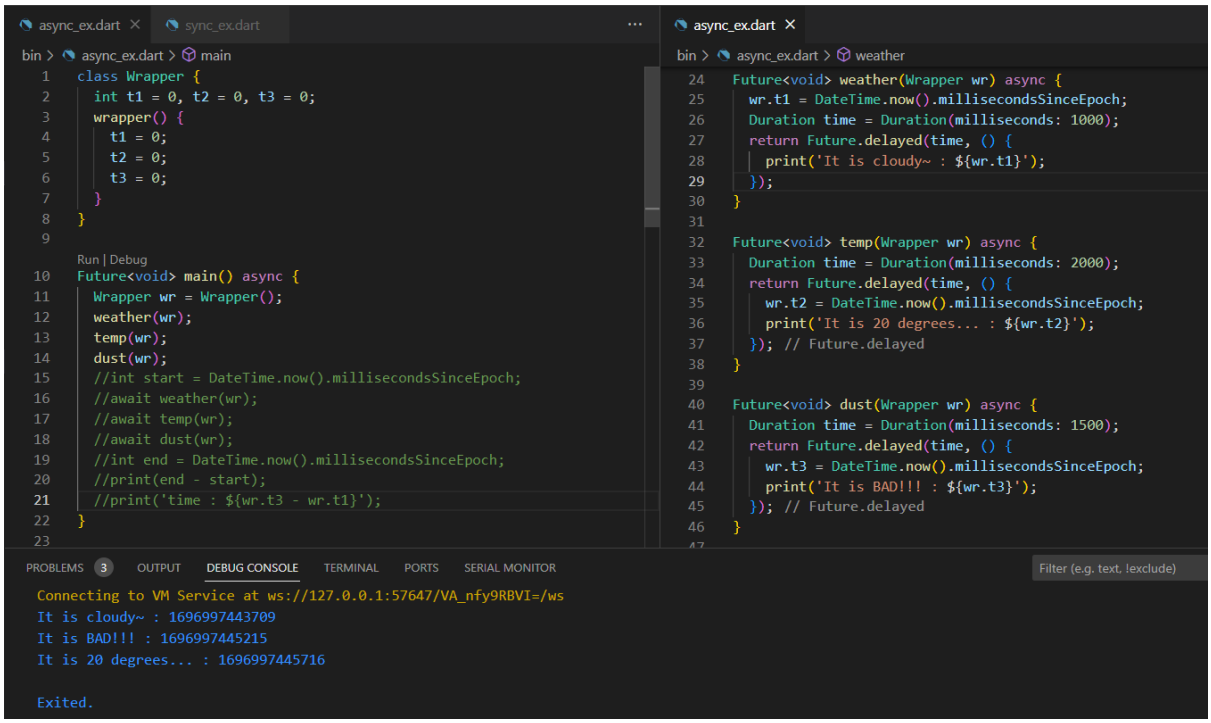
```
Connecting to VM Service at ws://127.0.0.1:57457/eIkaTwy8hFQ=/
It is cloudy
It is 20 degrees
It is bad
4542

Exited.
```

Async program → 2007 millisconds



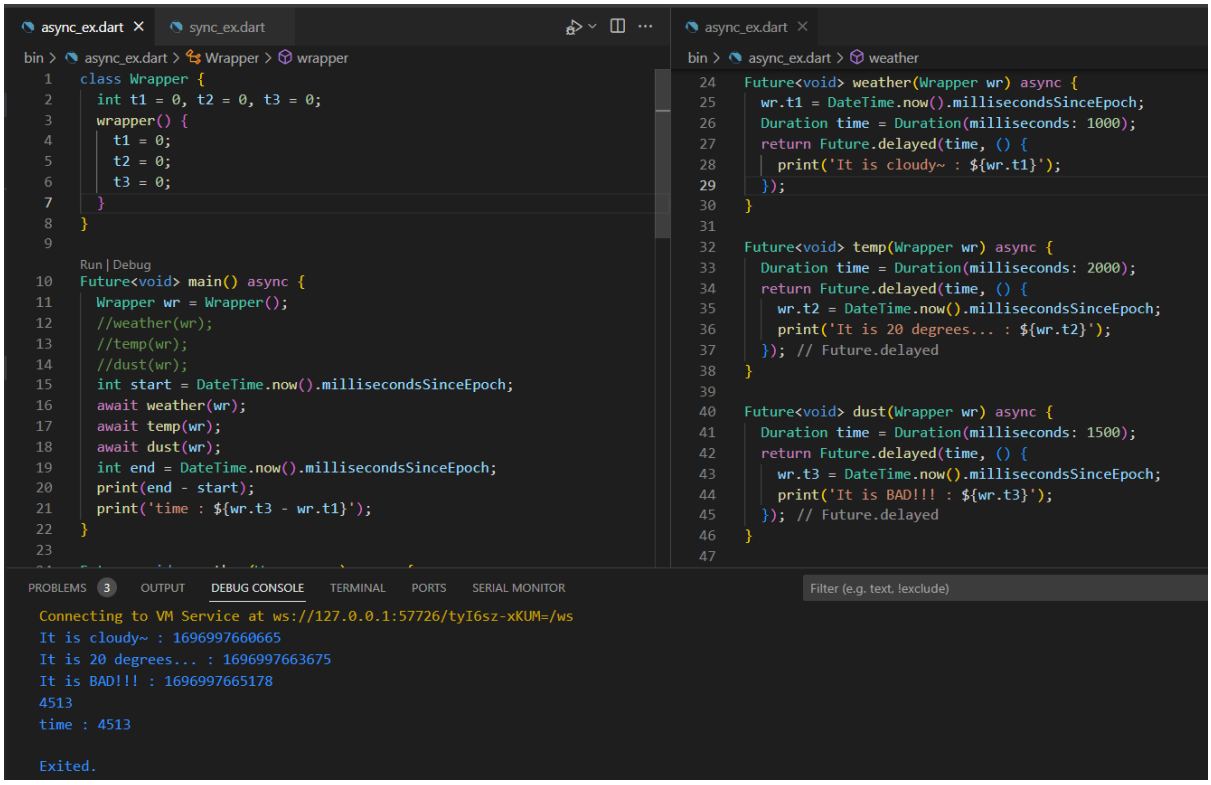
```
async_ex.dart
sync_ex.dart
bin > async_ex.dart > main
class Wrapper {
  int t1 = 0, t2 = 0, t3 = 0;
  wrapper() {
    t1 = 0;
    t2 = 0;
    t3 = 0;
  }
}

Run | Debug
Future<void> main() async {
  Wrapper wr = Wrapper();
  weather(wr);
  temp(wr);
  dust(wr);
  //int start = DateTime.now().millisecondsSinceEpoch;
  //await weather(wr);
  //await temp(wr);
  //await dust(wr);
  //int end = DateTime.now().millisecondsSinceEpoch;
  //print(end - start);
  //print('time : ${wr.t3 - wr.t1}');
}

async_ex.dart
bin > async_ex.dart > weather
Future<void> weather(Wrapper wr) async {
  wr.t1 = DateTime.now().millisecondsSinceEpoch;
  Duration time = Duration(milliseconds: 1000);
  return Future.delayed(time, () {
    print('It is cloudy~ : ${wr.t1}');
  });
}

Future<void> temp(Wrapper wr) async {
  Duration time = Duration(milliseconds: 2000);
  return Future.delayed(time, () {
    wr.t2 = DateTime.now().millisecondsSinceEpoch;
    print('It is 20 degrees... : ${wr.t2}');
  }); // Future.delayed
}

Future<void> dust(Wrapper wr) async {
  Duration time = Duration(milliseconds: 1500);
  return Future.delayed(time, () {
    wr.t3 = DateTime.now().millisecondsSinceEpoch;
    print('It is BAD!!! : ${wr.t3}');
  }); // Future.delayed
}

PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR
Filter (e.g. text, !exclude)

Connecting to VM Service at ws://127.0.0.1:57647/VA_nfy9RBVI=/ws
It is cloudy~ : 1696997443709
It is BAD!!! : 1696997445215
It is 20 degrees... : 1696997445716

Exited.
```

Async program → 2007 millisconds



```
async_ex.dart
sync_ex.dart
bin > async_ex.dart > main
class Wrapper {
  int t1 = 0, t2 = 0, t3 = 0;
  wrapper() {
    t1 = 0;
    t2 = 0;
    t3 = 0;
  }
}

Run | Debug
Future<void> main() async {
  Wrapper wr = Wrapper();
  weather(wr);
  temp(wr);
  dust(wr);
  //int start = DateTime.now().millisecondsSinceEpoch;
  //await weather(wr);
  //await temp(wr);
  //await dust(wr);
  //int end = DateTime.now().millisecondsSinceEpoch;
  //print(end - start);
  //print('time : ${wr.t3 - wr.t1}');
}

async_ex.dart
bin > async_ex.dart > weather
Future<void> weather(Wrapper wr) async {
  wr.t1 = DateTime.now().millisecondsSinceEpoch;
  Duration time = Duration(milliseconds: 1000);
  return Future.delayed(time, () {
    print('It is cloudy~ : ${wr.t1}');
  });
}

Future<void> temp(Wrapper wr) async {
  Duration time = Duration(milliseconds: 2000);
  return Future.delayed(time, () {
    wr.t2 = DateTime.now().millisecondsSinceEpoch;
    print('It is 20 degrees... : ${wr.t2}');
  }); // Future.delayed
}

Future<void> dust(Wrapper wr) async {
  Duration time = Duration(milliseconds: 1500);
  return Future.delayed(time, () {
    wr.t3 = DateTime.now().millisecondsSinceEpoch;
    print('It is BAD!!! : ${wr.t3}');
  }); // Future.delayed
}

PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR
Filter (e.g. text, !exclude)

Connecting to VM Service at ws://127.0.0.1:57647/VA_nfy9RBVI=/ws
It is cloudy~ : 1696997443709
It is BAD!!! : 1696997445215
It is 20 degrees... : 1696997445716

Exited.
```

Async~await program → 4513 millisconds

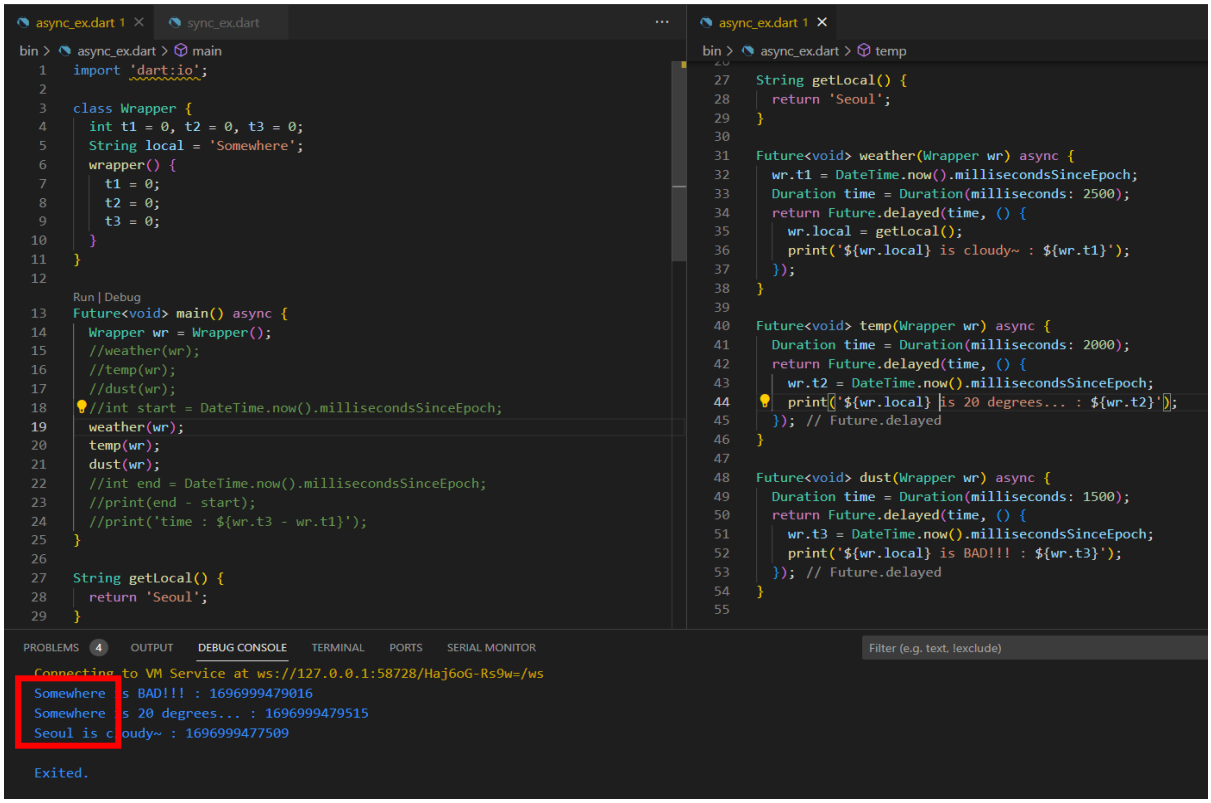


```
async_ex.dart
sync_ex.dart
bin > async_ex.dart > Wrapper > wrapper
class Wrapper {
  int t1 = 0, t2 = 0, t3 = 0;
  wrapper() {
    t1 = 0;
    t2 = 0;
    t3 = 0;
  }
}

Run | Debug
Future<void> main() async {
  Wrapper wr = Wrapper();
  //weather(wr);
  //temp(wr);
  //dust(wr);
  int start = DateTime.now().millisecondsSinceEpoch;
  await weather(wr);
  await temp(wr);
  await dust(wr);
  int end = DateTime.now().millisecondsSinceEpoch;
  print(end - start);
  print('time : ${wr.t3 - wr.t1}');
}
```

```
async_ex.dart
bin > async_ex.dart > weather
Future<void> weather(Wrapper wr) async {
  wr.t1 = DateTime.now().millisecondsSinceEpoch;
  Duration time = Duration(milliseconds: 1000);
  return Future.delayed(time, () {
    print('It is cloudy~ : ${wr.t1}');
  });
}

Future<void> temp(Wrapper wr) async {
  Duration time = Duration(milliseconds: 2000);
  return Future.delayed(time, () {
    wr.t2 = DateTime.now().millisecondsSinceEpoch;
    print('It is 20 degrees... : ${wr.t2}');
  }); // Future.delayed
}

Future<void> dust(Wrapper wr) async {
  Duration time = Duration(milliseconds: 1500);
  return Future.delayed(time, () {
    wr.t3 = DateTime.now().millisecondsSinceEpoch;
    print('It is BAD!!! : ${wr.t3}');
  }); // Future.delayed
}
```

```
PROBLEMS 3  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR
Filter (e.g. text, !exclude)

Connecting to VM Service at ws://127.0.0.1:57726/tyI6sz-xKUM=/ws
It is cloudy~ : 1696997660665
It is 20 degrees... : 1696997663675
It is BAD!!! : 1696997665178
4513
time : 4513

Exited.
```

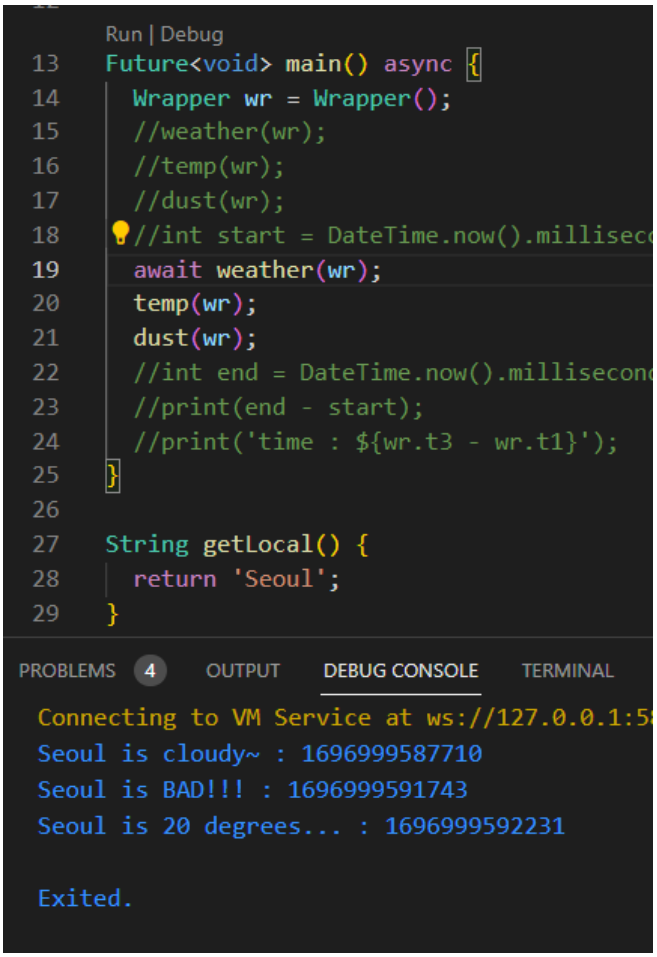
Async 진행에서 Sync 진행이 필요한 경우 await~



```
async_ex.dart 1 | sync_ex.dart
bin > async_ex.dart > main
1  import 'dart:io';
2
3  class Wrapper {
4    int t1 = 0, t2 = 0, t3 = 0;
5    String local = 'Somewhere';
6    wrapper() {
7      t1 = 0;
8      t2 = 0;
9      t3 = 0;
10   }
11 }
12
   Run | Debug
13 Future<void> main() async {
14   Wrapper wr = Wrapper();
15   //weather(wr);
16   //temp(wr);
17   //dust(wr);
18   //int start = DateTime.now().millisecondsSinceEpoch;
19   weather(wr);
20   temp(wr);
21   dust(wr);
22   //int end = DateTime.now().millisecondsSinceEpoch;
23   //print(end - start);
24   //print('time : ${wr.t3 - wr.t1}');
25 }
26
27 String getLocal() {
28   return 'Seoul';
29 }
```

```
async_ex.dart 1
bin > async_ex.dart > temp
27 String getLocal() {
28   return 'Seoul';
29 }
30
31 Future<void> weather(Wrapper wr) async {
32   wr.t1 = DateTime.now().millisecondsSinceEpoch;
33   Duration time = Duration(milliseconds: 2500);
34   return Future.delayed(time, () {
35     wr.local = getLocal();
36     print('${wr.local} is cloudy~ : ${wr.t1}');
37   });
38 }
39
40 Future<void> temp(Wrapper wr) async {
41   Duration time = Duration(milliseconds: 2000);
42   return Future.delayed(time, () {
43     wr.t2 = DateTime.now().millisecondsSinceEpoch;
44     print('${wr.local} is 20 degrees... : ${wr.t2}');
45   }); // Future.delayed
46 }
47
48 Future<void> dust(Wrapper wr) async {
49   Duration time = Duration(milliseconds: 1500);
50   return Future.delayed(time, () {
51     wr.t3 = DateTime.now().millisecondsSinceEpoch;
52     print('${wr.local} is BAD!!! : ${wr.t3}');
53   }); // Future.delayed
54 }
55
```

```
PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   SERIAL MONITOR
Filter (e.g. text, !exclude)

Connecting to VM Service at ws://127.0.0.1:58728/Haj6oG-Rs9w=/ws
Somewhere is BAD!!! : 1696999479016
Somewhere is 20 degrees... : 1696999479515
Seoul is cloudy~ : 1696999477509

Exited.
```



```
   Run | Debug
13 Future<void> main() async {
14   Wrapper wr = Wrapper();
15   //weather(wr);
16   //temp(wr);
17   //dust(wr);
18   //int start = DateTime.now().millisec
19   await weather(wr);
20   temp(wr);
21   dust(wr);
22   //int end = DateTime.now().millisecon
23   //print(end - start);
24   //print('time : ${wr.t3 - wr.t1}');
25 }
26
27 String getLocal() {
28   return 'Seoul';
29 }
```

```
PROBLEMS 4   OUTPUT   DEBUG CONSOLE   TERMINAL

Connecting to VM Service at ws://127.0.0.1:5
Seoul is cloudy~ : 1696999587710
Seoul is BAD!!! : 1696999591743
Seoul is 20 degrees... : 1696999592231

Exited.
```

Execution flow with async and await

An async function runs synchronously until the first await keyword. This means that within an async function body, all synchronous code before the first await keyword executes immediately.

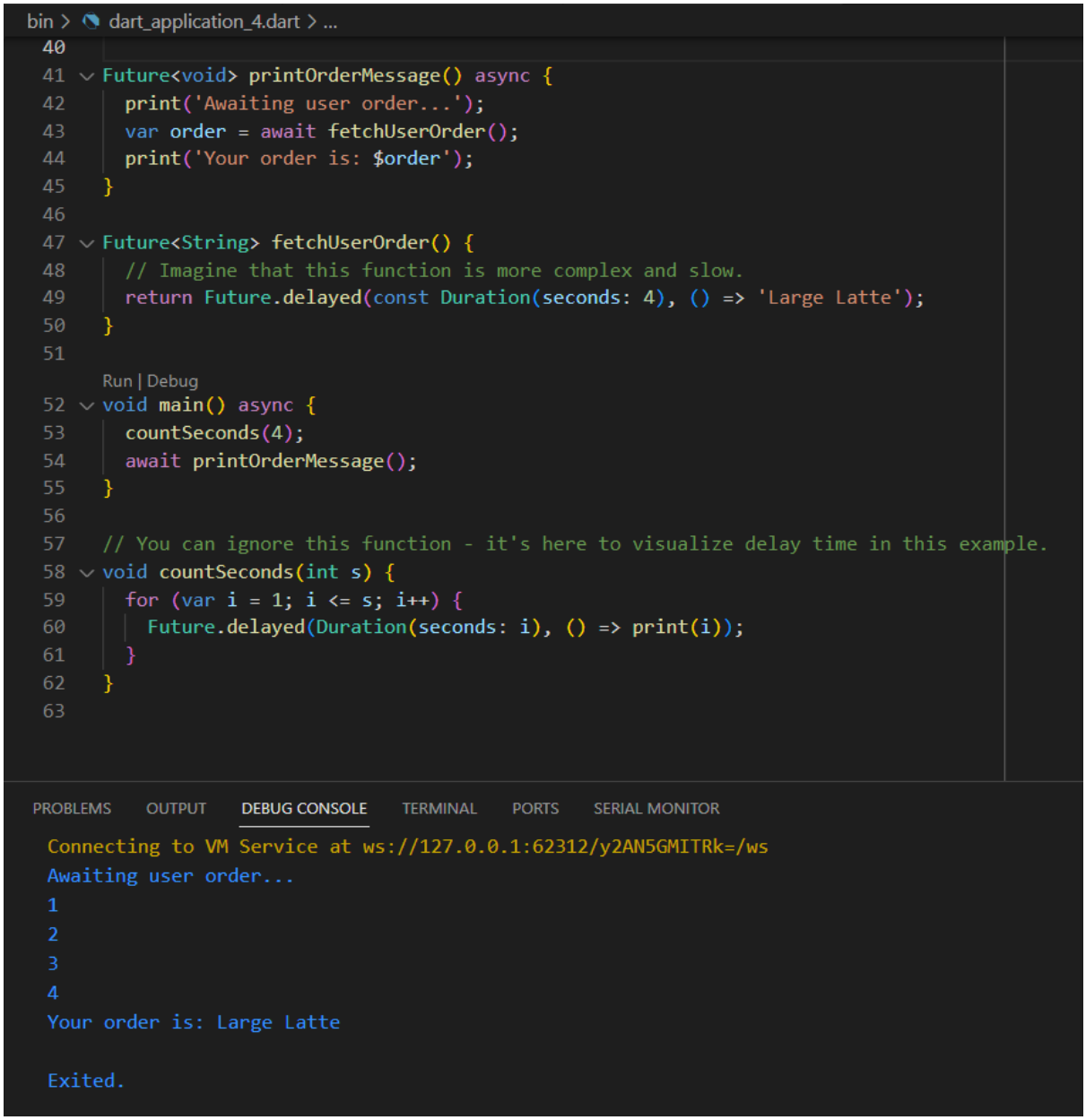
Ex1)

```dart
// Example: synchronous functions
String createOrderMessage() {
  var order = fetchUserOrder();
  return 'Your order is: $order';
}

Future<String> fetchUserOrder() =>
    // Imagine that this function is more complex and slow.
    Future.delayed(
      const Duration(seconds: 2),
      () => 'Large Latte',
    ); // Future.delayed

Run | Debug
void main() {
  print(createOrderMessage());
}
```

Ex2)

```dart
// Example: asynchronous functions
Future<String> createOrderMessage() async {
  var order = await fetchUserOrder();
  return 'Your order is: $order';
}

Future<String> fetchUserOrder() =>
    // Imagine that this function is
    // more complex and slow.
    Future.delayed(
      const Duration(seconds: 2),
      () => 'Large Latte',
    ); // Future.delayed

Run | Debug
Future<void> main() async {
  print('Fetching user order...');
  print(await createOrderMessage());
}
```

Ex3)



```
bin > dart_application_4.dart > ...
40
41  Future<void> printOrderMessage() async {
42    print('Awaiting user order...');
43    var order = await fetchUserOrder();
44    print('Your order is: $order');
45  }
46
47  Future<String> fetchUserOrder() {
48    // Imagine that this function is more complex and slow.
49    return Future.delayed(const Duration(seconds: 4), () => 'Large Latte');
50  }
51
    Run | Debug
52  void main() async {
53    countSeconds(4);
54    await printOrderMessage();
55  }
56
57  // You can ignore this function - it's here to visualize delay time in this example.
58  void countSeconds(int s) {
59    for (var i = 1; i <= s; i++) {
60      Future.delayed(Duration(seconds: i), () => print(i));
61    }
62  }
63
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR
Connecting to VM Service at ws://127.0.0.1:62312/y2AN5GMITRk=/ws
Awaiting user order...
1
2
3
4
Your order is: Large Latte

Exited.
```

JSON, sample code #5

JSON (JavaScript Object Notation) is a lightweight data-interchange format

https://www.json.org/json-en.html

https://ko.wikipedia.org/wiki/JSON

배열(Array) [편집]

배열은 대괄호[]로 나타낸다. 배열의 각 요소는 기본 자료형이거나 배열, 객체이다. 각 요소들은 쉼표(,)로 구별된다. 각 요소가 나타나는 순서에 의미가 있다.

```
[10, {"v": 20}, [30, "마흔"]]
```

객체(Object) [편집]

객체는 이름/값 쌍의 집합으로, 중괄호{}를 사용한다. 이름은 문자열이기 때문에 반드시 따옴표를 하며, 값은 기본 자료형이다. 각 쌍들은 쉼표(,)로 구별된다. 각 쌍이 나오는 순서는 의미가 없다.

```
{"name2": 50, "name3": "값3", "name1": true}
```

JSON 메시지 단위는 배열이나 객체이다. 위의 두 예는 JSON 메시지가 될 수 있다.

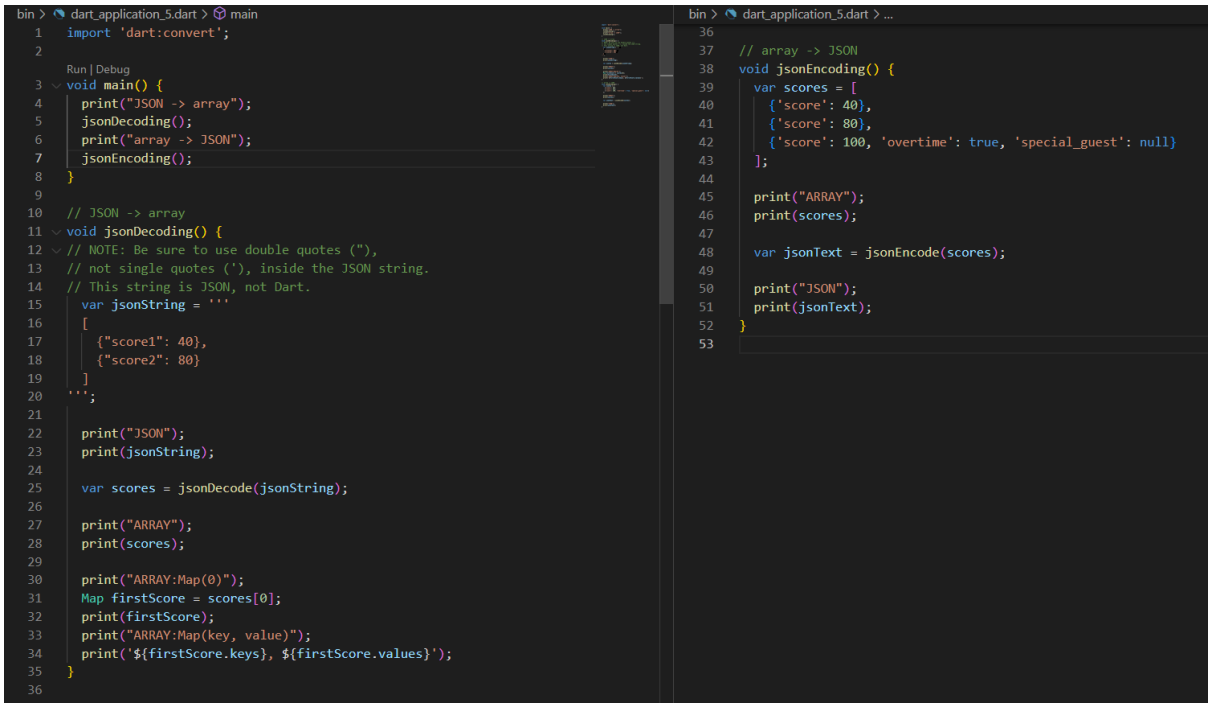
예제 [편집]

다음은 한 사람에 관한 정보를 갖는 JSON 객체이다.

키-값 쌍(이름:값)의 패턴으로 표현된다.

```
{
   "이름": "홍길동",
   "나이": 55,
   "성별": "남",
   "주소": "서울특별시 양천구 목동",
   "특기": ["검술", "코딩"],
   "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},
   "회사": "경기 수원시 팔달구 우만동"
}
```

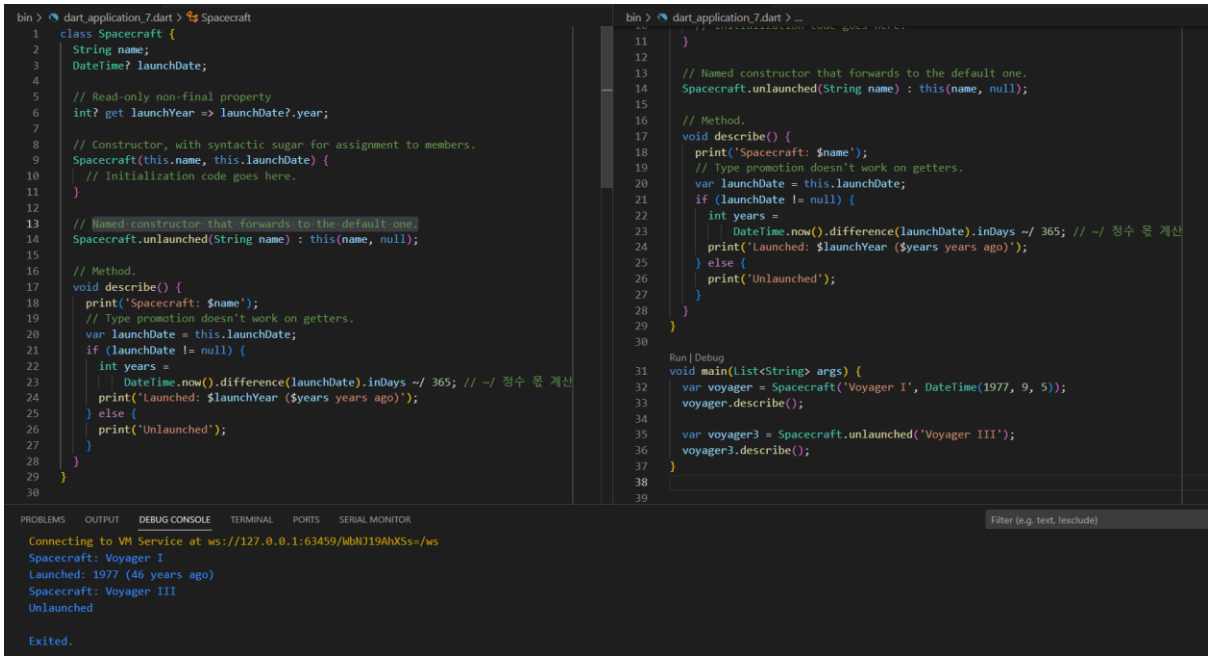
https://dart.dev/guides/libraries/library-tour#dartconvert---decoding-and-encoding-json-utf-8-and-more



```dart
import 'dart:convert';

void main() {
  print("JSON -> array");
  jsonDecoding();
  print("array -> JSON");
  jsonEncoding();
}

// JSON -> array
void jsonDecoding() {
// NOTE: Be sure to use double quotes ("),
// not single quotes ('), inside the JSON string.
// This string is JSON, not Dart.
  var jsonString = '''
  [
    {"score1": 40},
    {"score2": 80}
  ]
''';

  print("JSON");
  print(jsonString);

  var scores = jsonDecode(jsonString);

  print("ARRAY");
  print(scores);

  print("ARRAY:Map(0)");
  Map firstScore = scores[0];
  print(firstScore);
  print("ARRAY:Map(key, value)");
  print('${firstScore.keys}, ${firstScore.values}');
}
```

```dart
// array -> JSON
void jsonEncoding() {
  var scores = [
    {'score': 40},
    {'score': 80},
    {'score': 100, 'overtime': true, 'special_guest': null}
  ];

  print("ARRAY");
  print(scores);

  var jsonText = jsonEncode(scores);

  print("JSON");
  print(jsonText);
}
```

Classes, sample code #6



```dart
class Spacecraft {
  String name;
  DateTime? launchDate;

  // Read-only non-final property
  int? get launchYear => launchDate?.year;

  // Constructor, with syntactic sugar for assignment to members.
  Spacecraft(this.name, this.launchDate) {
    // Initialization code goes here.
  }

  // Named constructor that forwards to the default one.
  Spacecraft.unlaunched(String name) : this(name, null);

  // Method.
  void describe() {
    print('Spacecraft: $name');
    // Type promotion doesn't work on getters.
    var launchDate = this.launchDate;
    if (launchDate != null) {
      int years =
          DateTime.now().difference(launchDate).inDays ~/ 365; // ~/ 정수 몫 계산
      print('Launched: $launchYear ($years years ago)');
    } else {
      print('Unlaunched');
    }
  }
}

Run | Debug
void main(List<String> args) {
  var voyager = Spacecraft('Voyager I', DateTime(1977, 9, 5));
  voyager.describe();

  var voyager3 = Spacecraft.unlaunched('Voyager III');
  voyager3.describe();
}
```

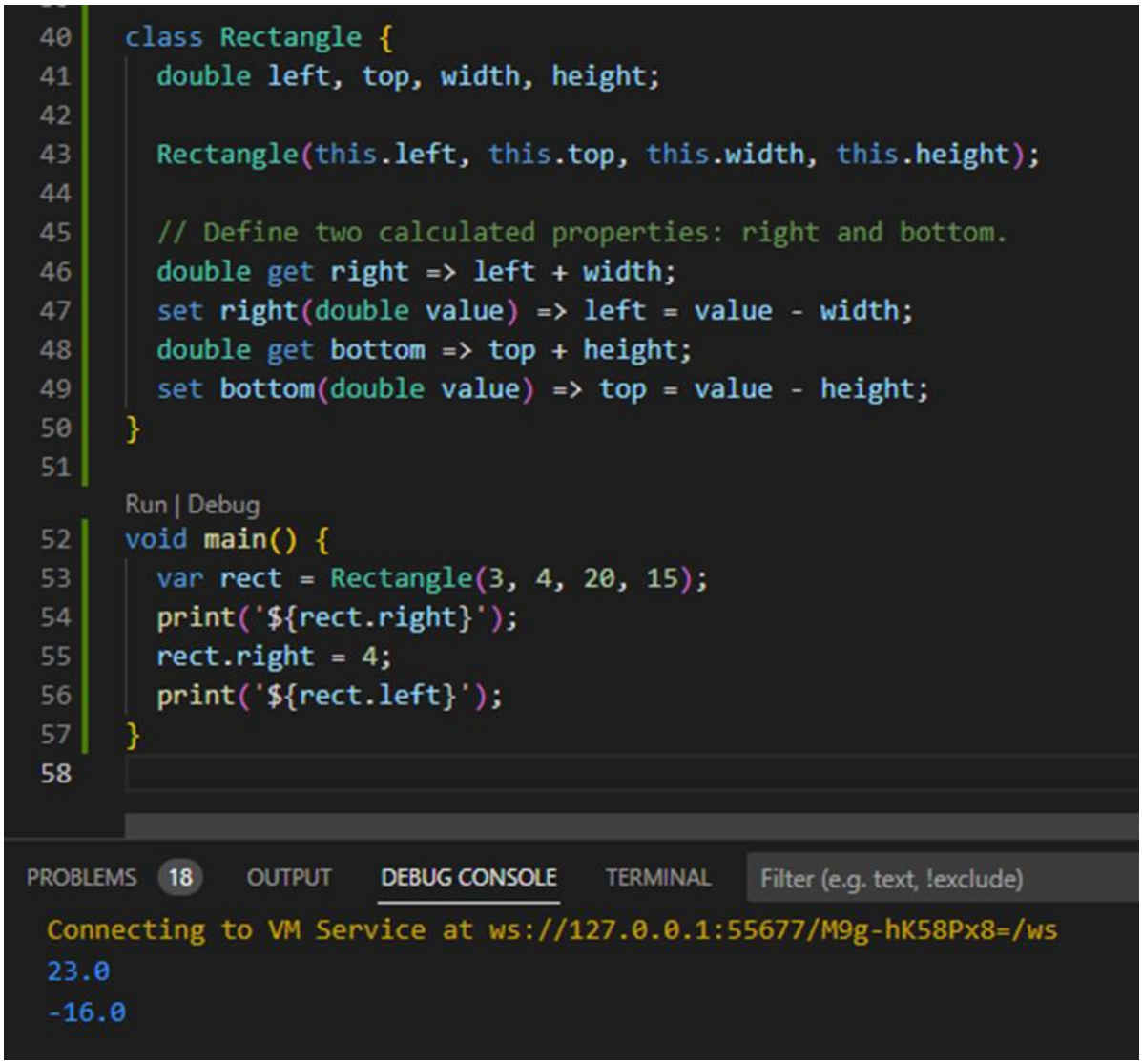
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SERIAL MONITOR
Connecting to VM Service at ws://127.0.0.1:63459/WbNJ19AhXSs=/ws
Spacecraft: Voyager I
Launched: 1977 (46 years ago)
Spacecraft: Voyager III
Unlaunched

Exited.
```

Getters and setters

게터 getter 와 세터 setter 함수는 일반적으로 어떤 데이터를 가져오거나 변경하는 함수를 의미

get 예약어를 추가한 함수는 데이터를 가져오는 게터

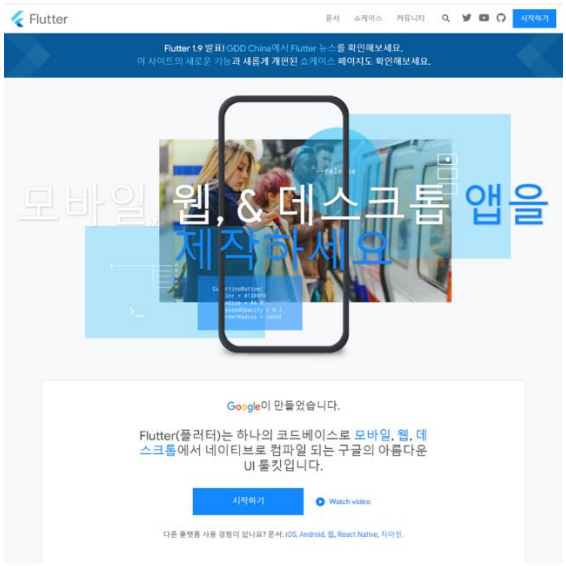
set 예약어를 추가한 함수는 데이터를 변경하는 세터



5. Flutter

5-1. Flutter?!

https://flutter.dev/



	Android	iOS
Native	Kotlin Java + AndroidStudio	Swift Objective-C + X code
Cross-platform	• React Native by Meta(Facebook) • Flutter by Google	

	React Native	Flutter
출시	2015.3.26	2017.5
Language	Javascript, Java Python, C++ Objective-C	Dart, C/C++

모바일 앱 종류와 특징 – 네이티브 앱, 크로스 플랫폼, 웹 앱, 하이브리드 앱

https://hongong.hanbit.co.kr/%EB%AA%A8%EB%B0%94%EC%9D%BC-%EC%95%B1-%EC%A2%85%EB%A5%98%EB%84%A4%EC%9D%B4%ED%8B%B0%EB%B8%8C-%EC%95%B1-%ED%81%AC%EB%A1%9C%EC%8A%A4-%ED%94%8C%EB%9E%AB%ED%8F%BC-%EC%9B%B9-%EC%95%B1-%ED%95%98/

네이티브 앱: 플랫폼별 애플리케이션 맞춤 제작

네이티브 앱은 안드로이드 스마트폰에서는 안드로이드 방식으로, 아이폰에서는 iOS 방식으로 개발되는 앱이기 때문에 각 스마트폰의 모든 기능을 사용할 수 있고 성능 또한 최대치로 활용할 수 있습니다. 작동하는 기기에 최적화된 형태로 개발되기 때문이죠. 그래서 카메라나 GPS, 블루투스 등 기기 내 장치들을 세밀히 다루거나, 영상 편집 앱과 같이 성능을 최대한 사용해야 하는 앱은 네이티브 앱으로 만드는 것이 좋아요.

크로스 플랫폼: 같은 소스 코드로 안드로이드 버전과 iOS 버전을 한 번에

크로스 플랫폼 프레임워크는 주로 한정된 자원을 가진 기기에서 비교적 단순한 기능으로 이뤄진 앱을 제작하는 데 사용합니다. 예를 들어 스케줄 관리 앱이나 쇼핑앱, 사진 보정 앱과 같이 캘린더, 카메라 등 스마트폰의 기본 기능들을 활용하면서도 높은 성능을 요구하지 않는 앱을 개발하기에 적합합니다.

웹 앱: 모바일 앱의 모습을 한 웹사이트

스마트폰에 설치된 브라우저에서 동작하는 웹사이트이기 때문에 웹 앱으로 할 수 있는 일은 '브라우저로 할 수 있는' 기능에 한정되죠

웹 앱 예시

하이브리드 앱: 네이티브 앱의 장점 + 웹 앱의 장점

하이브리드 앱의 원리는 간단합니다. 네이티브 또는 크로스 플랫폼 방식으로 앱을 만들되 화면 안에 요소를 직접 만들지 않고, 브라우저 역할을 하는 웹 뷰라는 요소를 만들어 대신 웹 화면을 띄웁니다. 그러면 웹 뷰는 지정된 주소로 접속해서 웹 앱 형태로 만들어진 웹사이트를 화면에 보여줍니다. 사용자는 이 웹 뷰를 통해 웹사이트로 만들어진 기능을 앱 기능처럼 사용합니다. 그리고 웹사이트에서 제공할 수 없는 하드웨어 기능인 카메라나 푸시 알림 같은 기능은 네이티브 단에서 자체적으로 수행하도록 하는 것입니다.

5-2. IDE 설치

https://docs.flutter.dev/get-started/install

Android 스튜디오 / IntelliJ　비주얼 스튜디오 코드

Android 스튜디오 설치

Android 스튜디오는 Flutter에 맞는 완벽한 통합 IDE 경험을 제공합니다.

- Android 스튜디오, 3.0 이상 버전

또는, IntelliJ를 사용할 수도 있습니다:

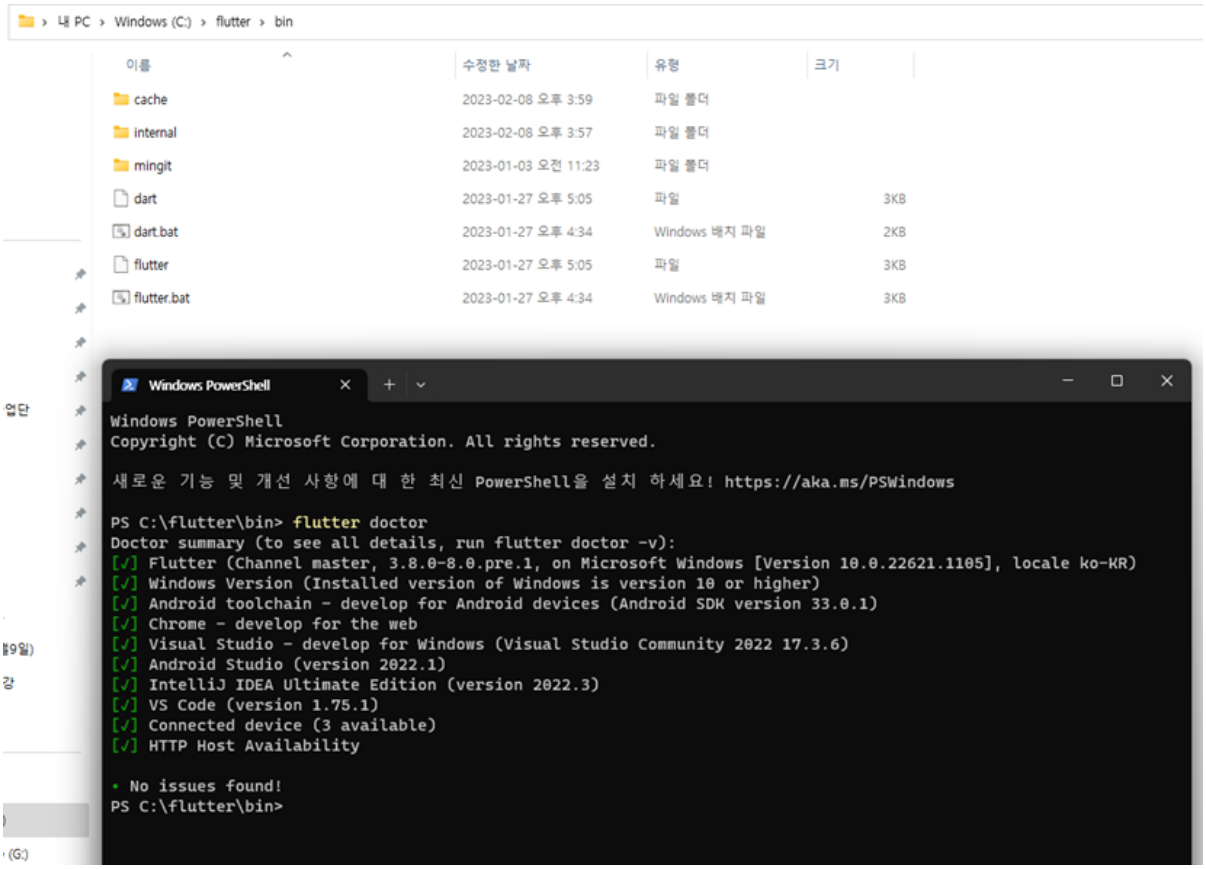
- IntelliJ IDEA Community, 2017.1 이상 버전
- IntelliJ IDEA Ultimate, 2017.1 이상 버전

Flutter와 Dart 플러그인 설치

설치를 위해서:

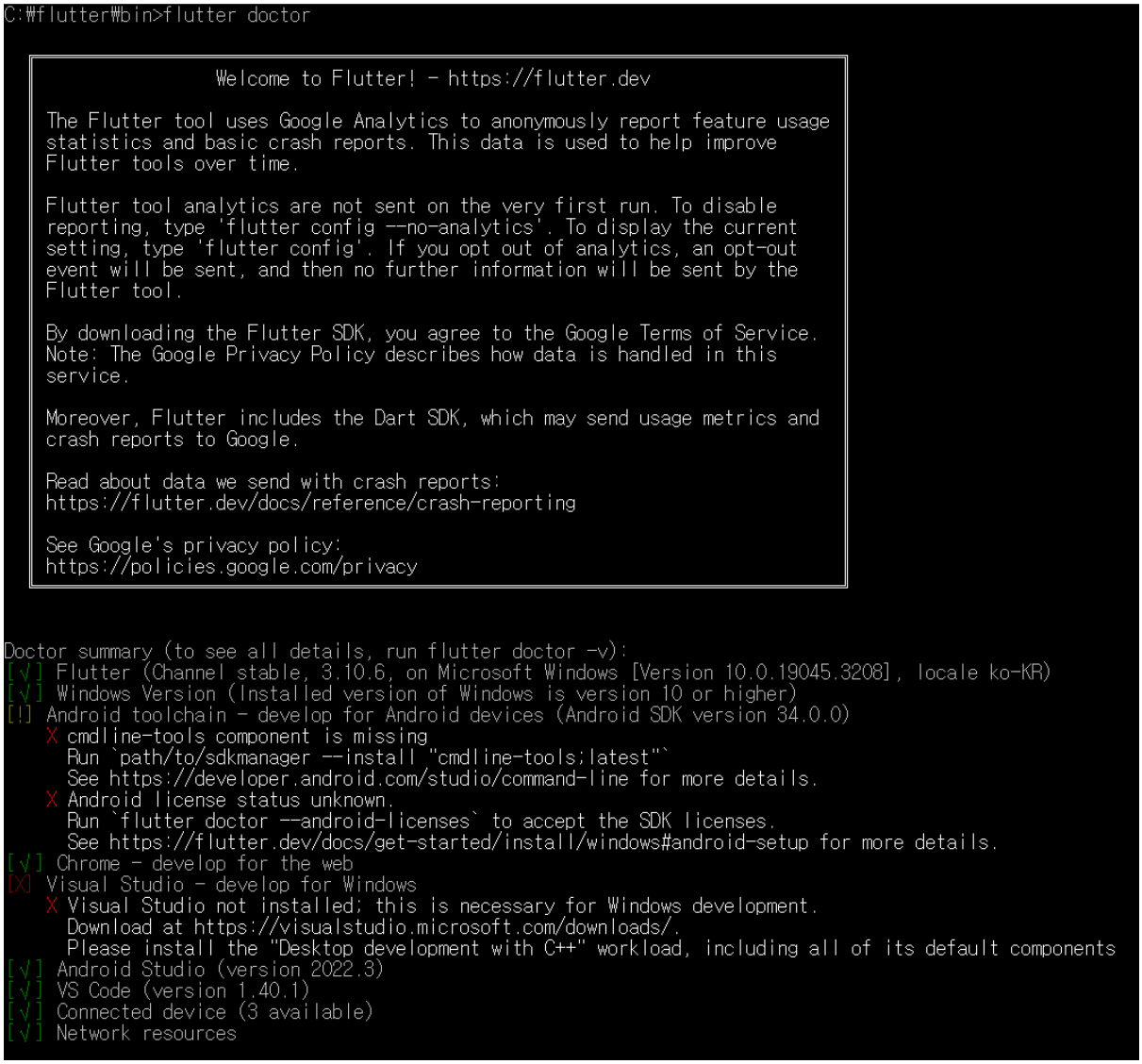
1. Android 스튜디오를 시작하세요.
2. 플러그인 preferences를 여세요 (맥OS에서는 **Preferences > Plugins**, 윈도우와 리눅스에서는 **File > Settings > Plugins**).
3. **Marketplace**를 선택하고, Flutter 플러그인을 선택하고 **설치**를 클릭하세요.
4. Dart 플러그인을 설치하라는 메시지가 나타나면 **Yes**를 클릭하세요.
5. 메시지가 나타나면 **Restart**를 클릭하세요.

Flutter SDK 설치 경로(./bin)에서 명령프롬프트 실행
C:₩flutter₩bin> flutter doctor

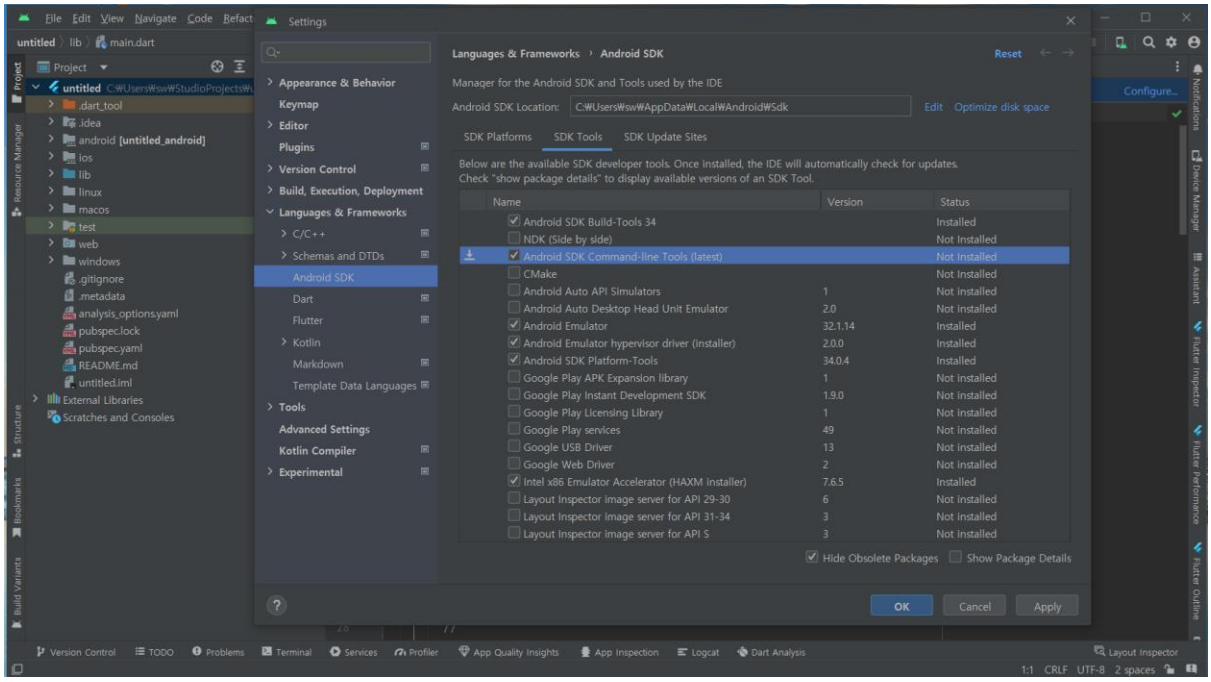


Tool chain issue

https://www.androidhuman.com/2021-06-02-flutter_android_license_noclassdeffound

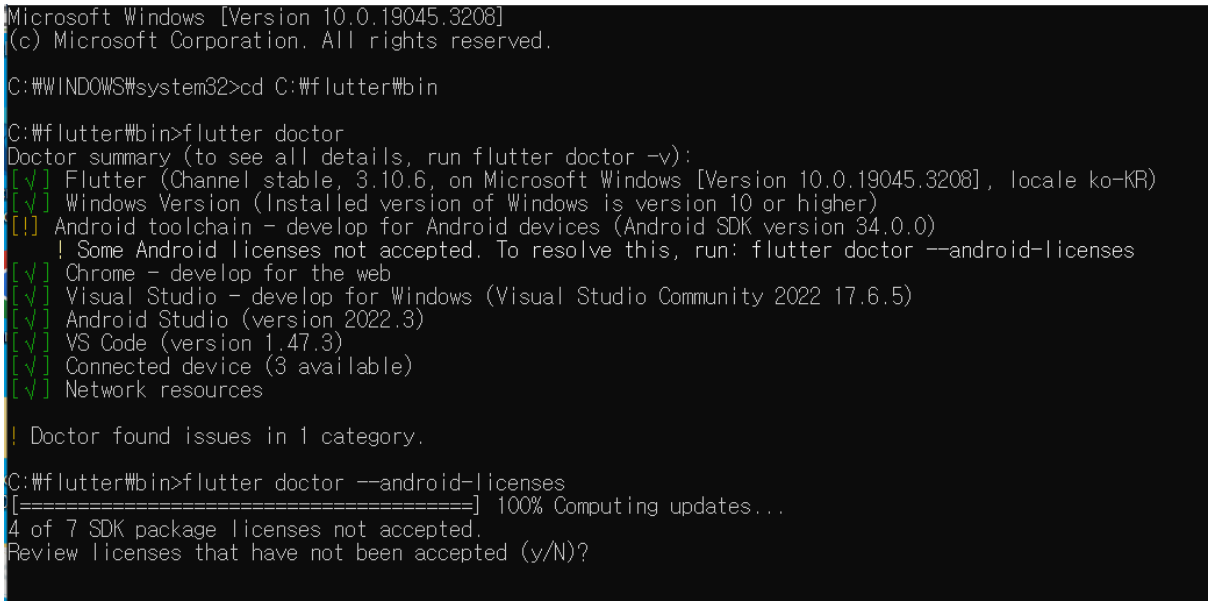


SDK Manager



Visual Studio - Desktop development with C++

flutter doctor --android-licenses



오류 - your sdk location contains non-ascii character

https://developeonmyown.tistory.com/2

오류 - flutter 설치시 android-sdk 를 못찾는 경우

https://highheat.tistory.com/732

[안드로이드] Caused by: org.gradle.api.internal.plugins.PluginApplicationException: Failed to apply plugin [id 'com.android.internal.application']

https://week-year.tistory.com/151

사용자 계정이 한글인 경우는 오류 발생→환경변수에 C:₩flutter₩bin 설정 필요함.

Android 스튜디오 / IntelliJ　비주얼 스튜디오 코드

VS Code 설치

VS Code는 Flutter 앱 실행 및 디버그를 지원하는 가벼운 에디터입니다.

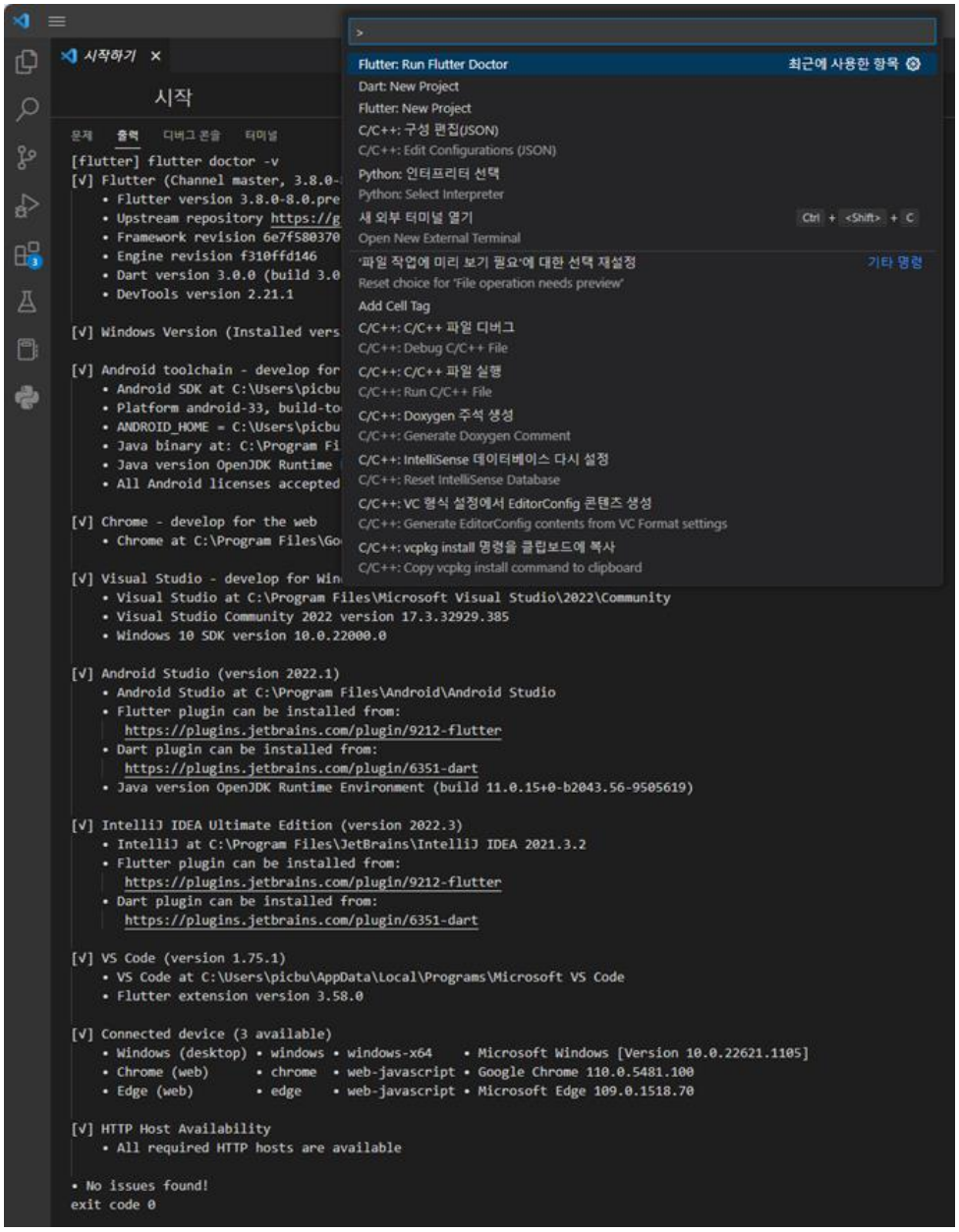
- VS Code, 최신 안정 버전

Flutter와 Dart 플러그인 설치

1. VS Code를 시작하세요.
2. **View > Command Palette...**를 호출하세요.
3. "install"를 입력하고, **Extensions: Install Extensions**를 선택하세요.
4. 확장 검색 필드에서 "flutter"를 입력하고, 리스트에서 **Flutter**를 선택한 뒤, **Install**을 클릭하세요. 이렇게하면 필수 Dart 플러그인도 함께 설치됩니다.
5. VS Code를 다시 로드하기 위해 **Reload to Activate**를 클릭하세요.
6. **Reload to Activate**을 눌러 VS Code를 다시 시작하세요.

Flutter Doctor로 설정 확인

1. **View > Command Palette...**를 호출하세요.
2. "doctor"를 입력하고, **Flutter: Run Flutter Doctor**를 선택하세요.
3. **OUTPUT** 창의 출력에 아무 문제가 없는지 확인하세요.


시작하기
시작
문제 출력 디버그 콘솔 터미널
[flutter] flutter doctor -v
[√] Flutter (Channel master, 3.8.0-
• Flutter version 3.8.0-8.0.pre
• Upstream repository https://g
• Framework revision 6e7f580370
• Engine revision f310ffd146
• Dart version 3.0.0 (build 3.0
• DevTools version 2.21.1
[√] Windows Version (Installed vers
[√] Android toolchain - develop for
• Android SDK at C:\Users\picbu
• Platform android-33, build-to
• ANDROID_HOME = C:\Users\picbu
• Java binary at: C:\Program Fi
• Java version OpenJDK Runtime
• All Android licenses accepted
[√] Chrome - develop for the web
• Chrome at C:\Program Files\Go
[√] Visual Studio - develop for Win
• Visual Studio at C:\Program Files\Microsoft Visual Studio\2022\Community
• Visual Studio Community 2022 version 17.3.32929.385
• Windows 10 SDK version 10.0.22000.0
[√] Android Studio (version 2022.1)
• Android Studio at C:\Program Files\Android\Android Studio
• Flutter plugin can be installed from:
https://plugins.jetbrains.com/plugin/9212-flutter
• Dart plugin can be installed from:
https://plugins.jetbrains.com/plugin/6351-dart
• Java version OpenJDK Runtime Environment (build 11.0.15+0-b2043.56-9505619)
[√] IntelliJ IDEA Ultimate Edition (version 2022.3)
• IntelliJ at C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.2
• Flutter plugin can be installed from:
https://plugins.jetbrains.com/plugin/9212-flutter
• Dart plugin can be installed from:
https://plugins.jetbrains.com/plugin/6351-dart
[√] VS Code (version 1.75.1)
• VS Code at C:\Users\picbu\AppData\Local\Programs\Microsoft VS Code
• Flutter extension version 3.58.0
[√] Connected device (3 available)
• Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.22621.1105]
• Chrome (web) • chrome • web-javascript • Google Chrome 110.0.5481.100
• Edge (web) • edge • web-javascript • Microsoft Edge 109.0.1518.70
[√] HTTP Host Availability
• All required HTTP hosts are available
• No issues found!
exit code 0
Flutter: Run Flutter Doctor
최근에 사용한 항목
Dart: New Project
Flutter: New Project
C/C++: 구성 편집(JSON)
C/C++: Edit Configurations (JSON)
Python: 인터프리터 선택
Python: Select Interpreter
새 외부 터미널 열기
Ctrl + <Shift> + C
Open New External Terminal
'파일 작업에 미리 보기 필요'에 대한 선택 재설정
기타 명령
Reset choice for 'File operation needs preview'
Add Cell Tag
C/C++: C/C++ 파일 디버그
C/C++: Debug C/C++ File
C/C++: C/C++ 파일 실행
C/C++: Run C/C++ File
C/C++: Doxygen 주석 생성
C/C++: Generate Doxygen Comment
C/C++: IntelliSense 데이터베이스 다시 설정
C/C++: Reset IntelliSense Database
C/C++: VC 형식 설정에서 EditorConfig 콘텐츠 생성
C/C++: Generate EditorConfig contents from VC Format settings
C/C++: vcpkg install 명령을 클립보드에 복사
C/C++: Copy vcpkg install command to clipboard

Mac

```
artcoding@artcoding ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.13.7, on macOS 14.0 23A344 darwin-arm64, locale
    ko-KR)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 15.0)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2022.3)
[✓] VS Code (version 1.83.1)
[✓] Connected device (2 available)
[✓] Network resources

• No issues found!
artcoding@artcoding ~ %
```

Flutter SDK 설치(https://flutter-ko.dev/get-started/install/macos)

terminal 에서 vim $HOME/.zshrc 를 입력합니다.

export PATH="$PATH:Documents/Work/SDK/flutter/bin" (내가 flutter 를 unzip 한 위치)

:wq 를 입력하고 저장 후 나가기 합니다.

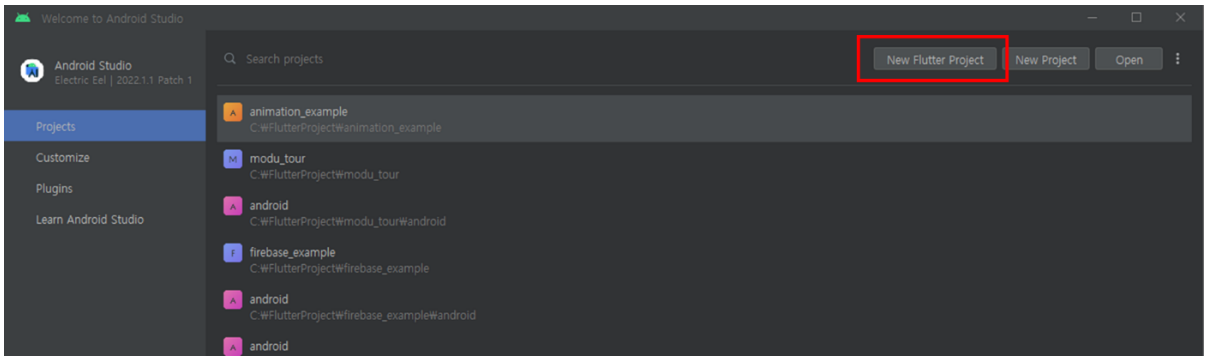
터미널을 열어 source $HOME/.zshrc 그리고 echo $PATH 를 차례로 입력합니다.

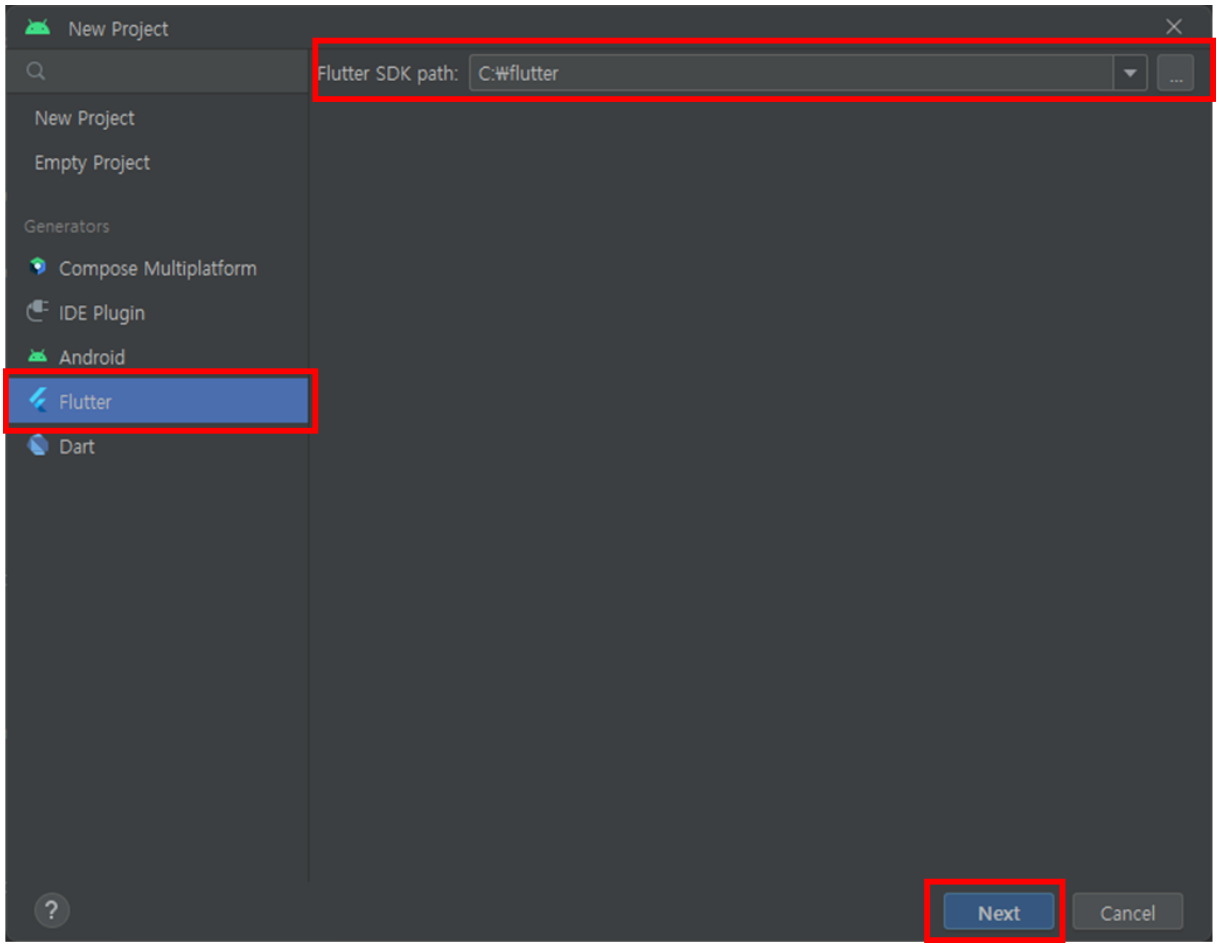
Verify : $which flutter

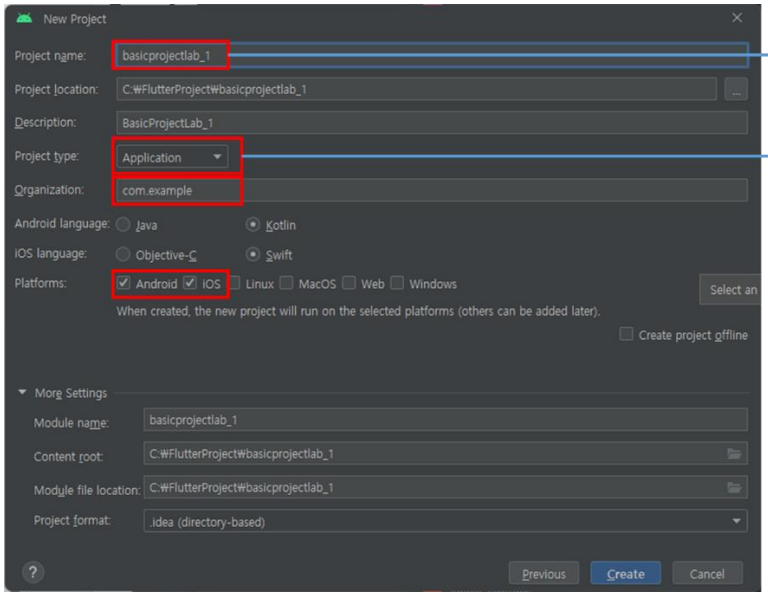
- Flutter 설치
- VSCode 설치
- Android Studio 설치
- JDK 설치
- XCode 설치
    - ✓ Simulator 설치
    - ✓ Xcode 를 실행한 뒤 상단바에서 Xcode - Open Developer Tool - Simulator 를 클릭하면 바로 가상 기기가 만들어집니다.
- Homebrew 설치 for cocoapod

5-3. Default Program : Counter

Android Studio 로 실습 진행



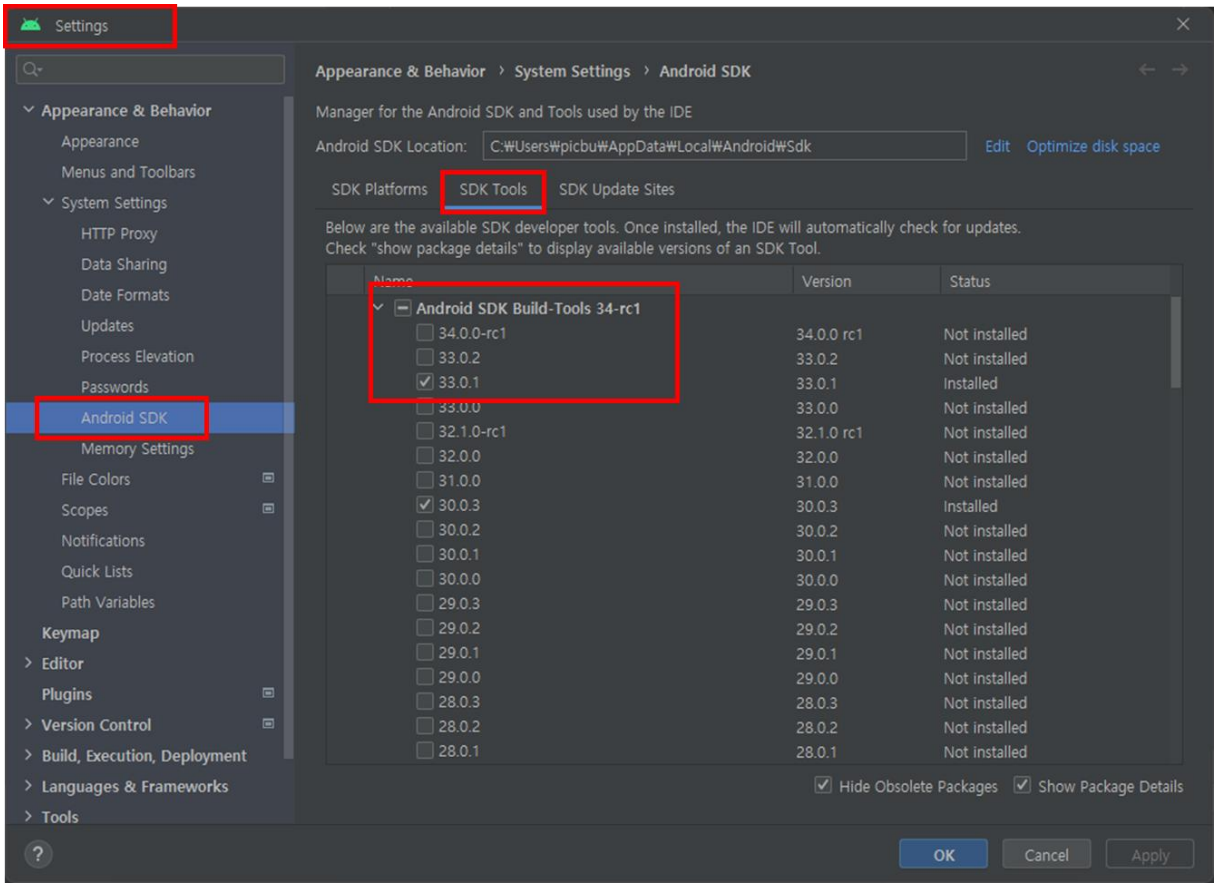


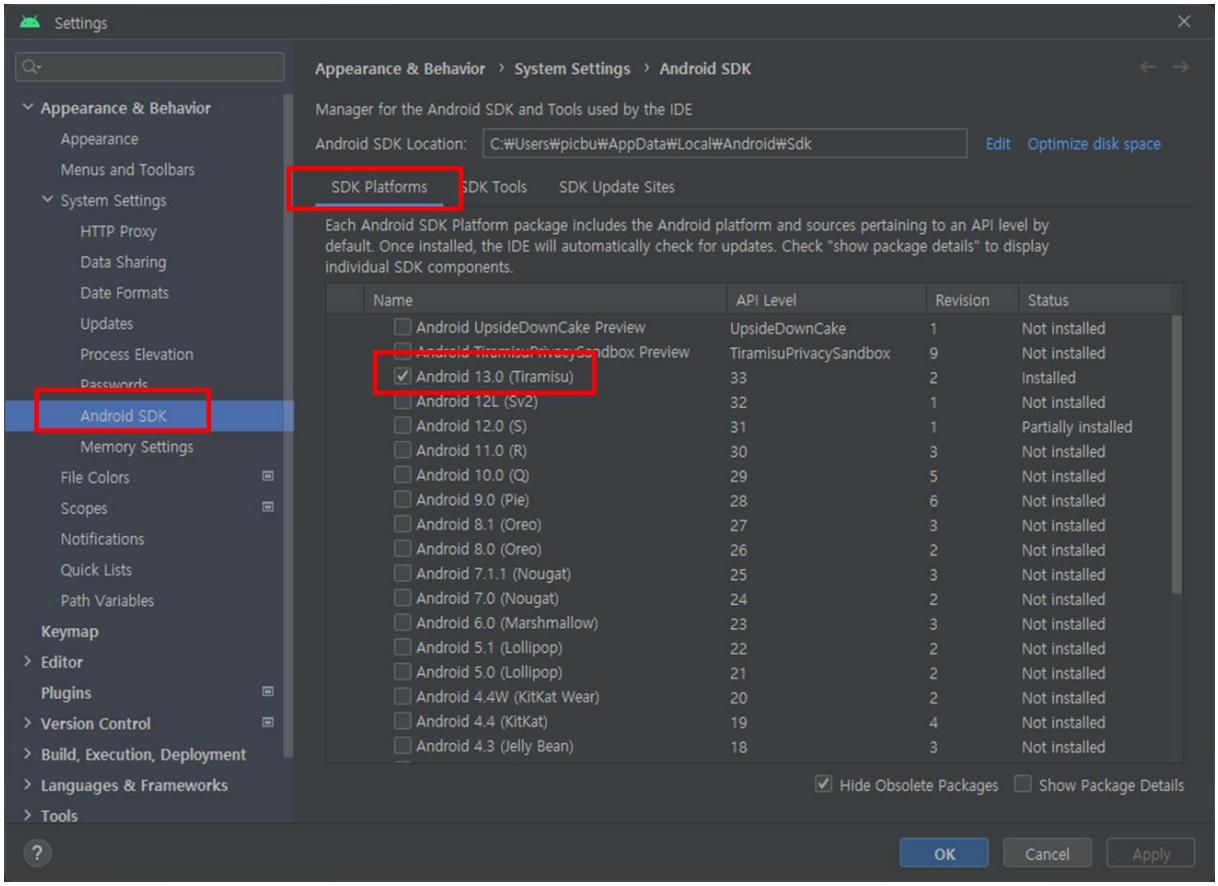


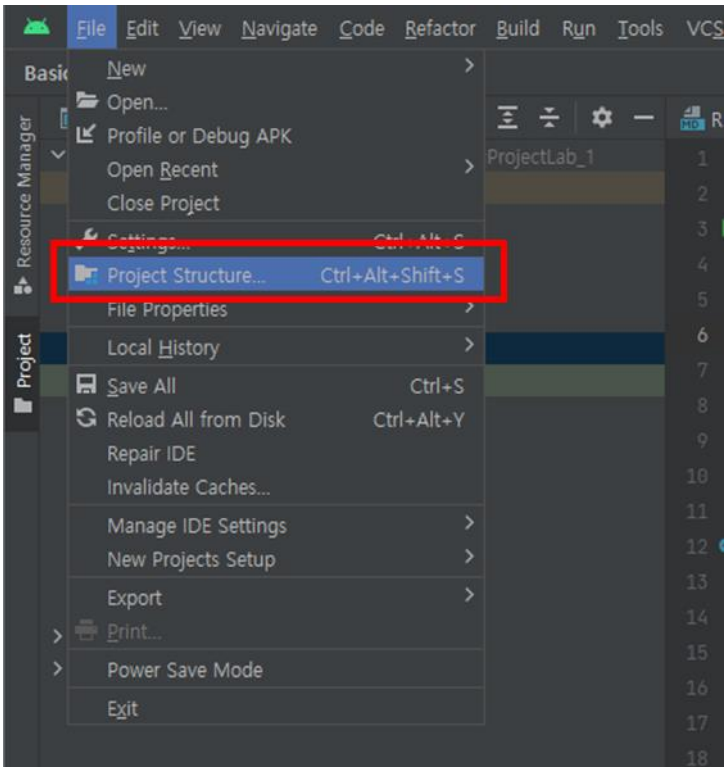
모두 소문자 또는 언더바,숫자 가능

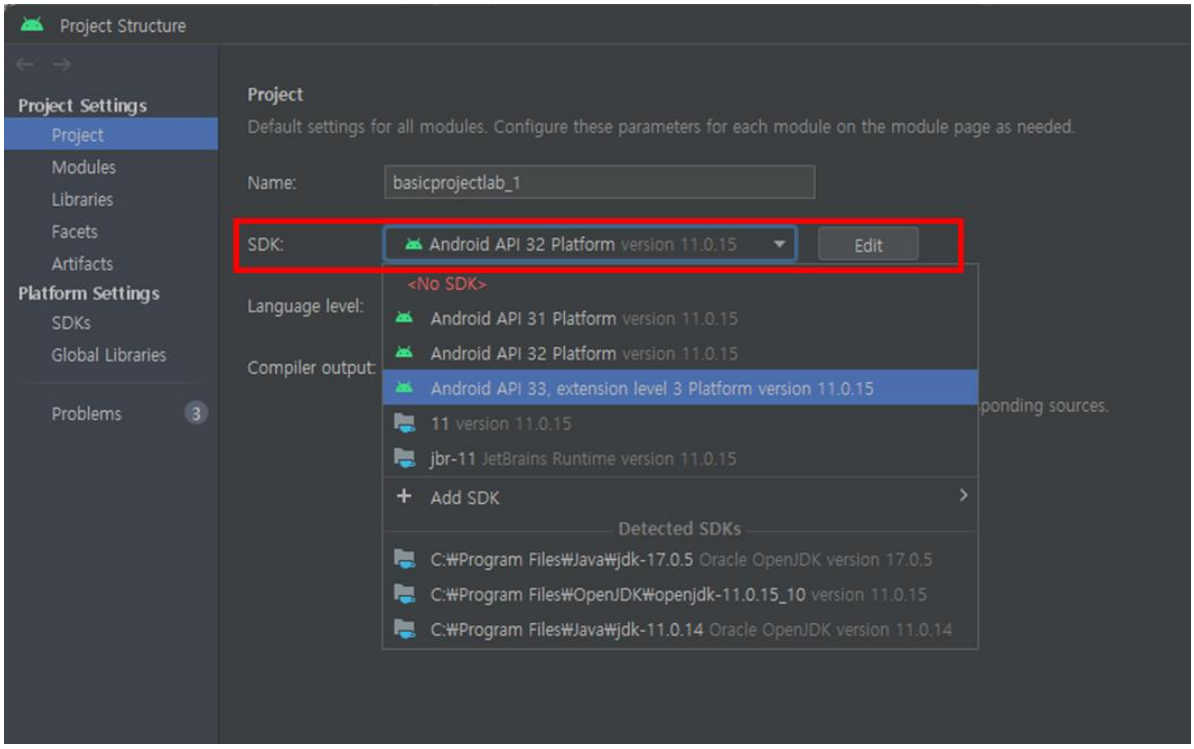
https://dart.dev/tools/pub/pubspec#name

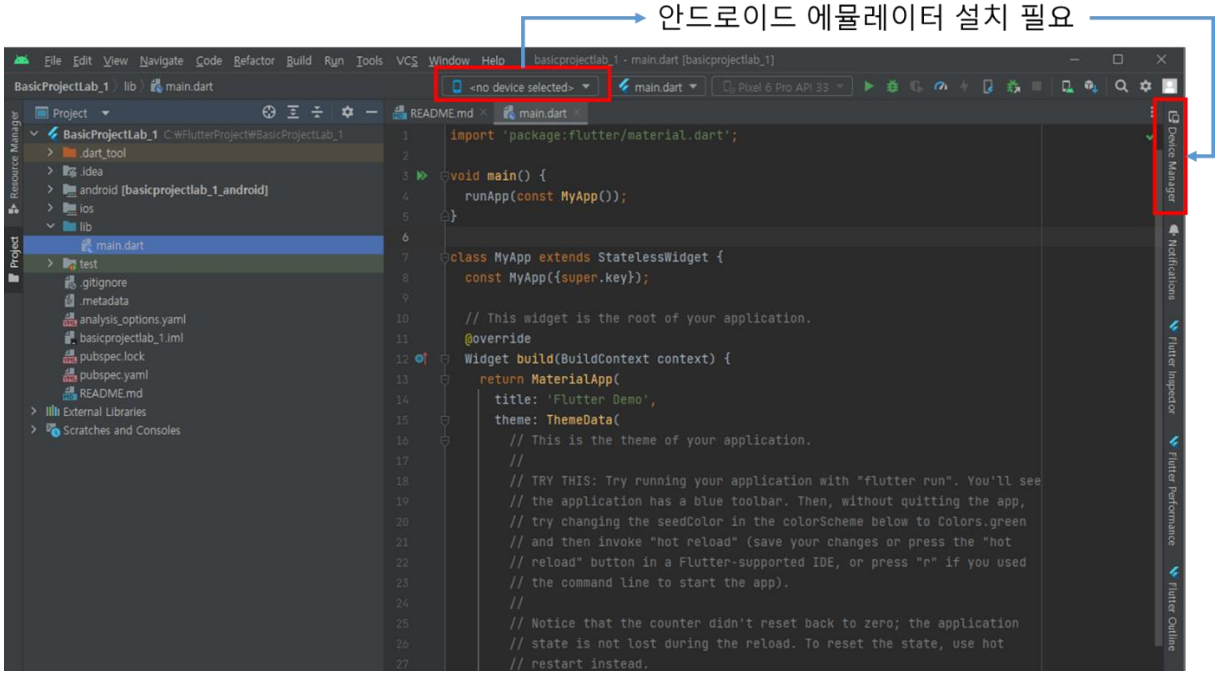
https://dart.dev/tools/pub/pubspec#name

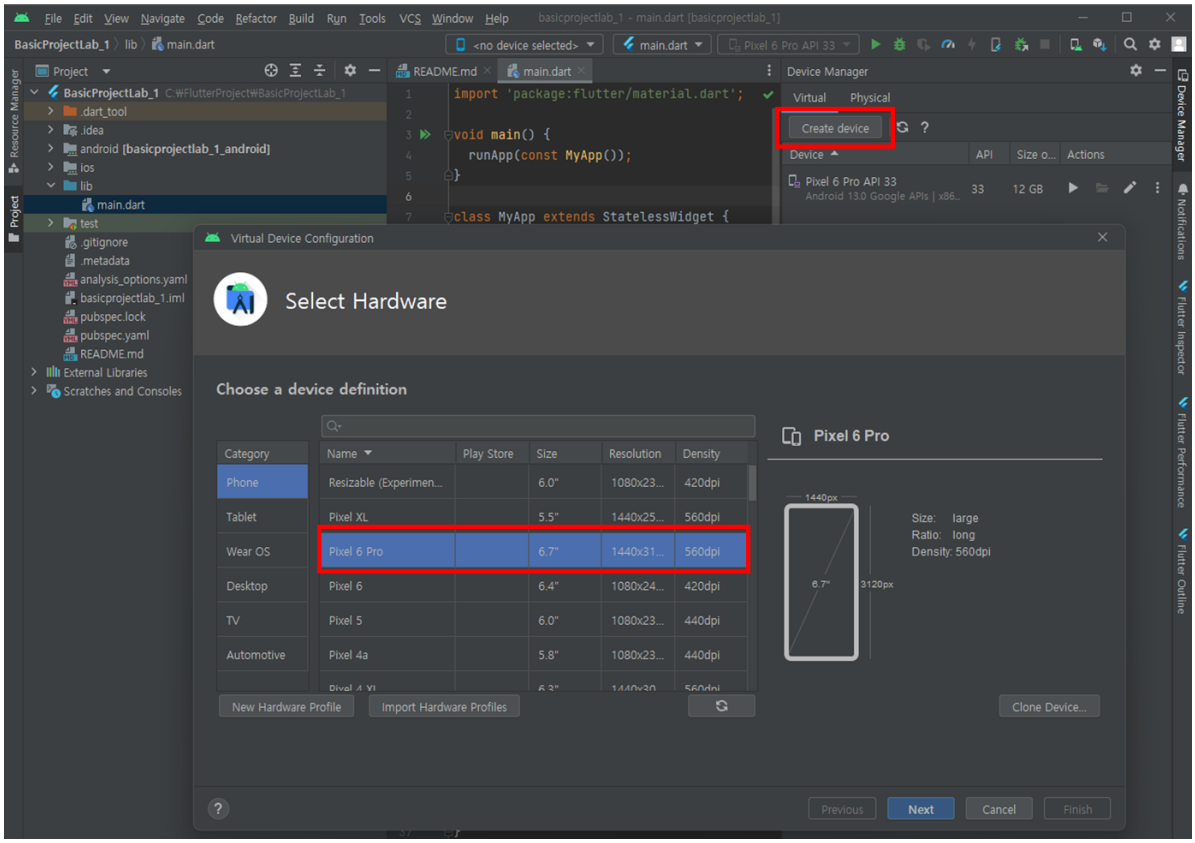


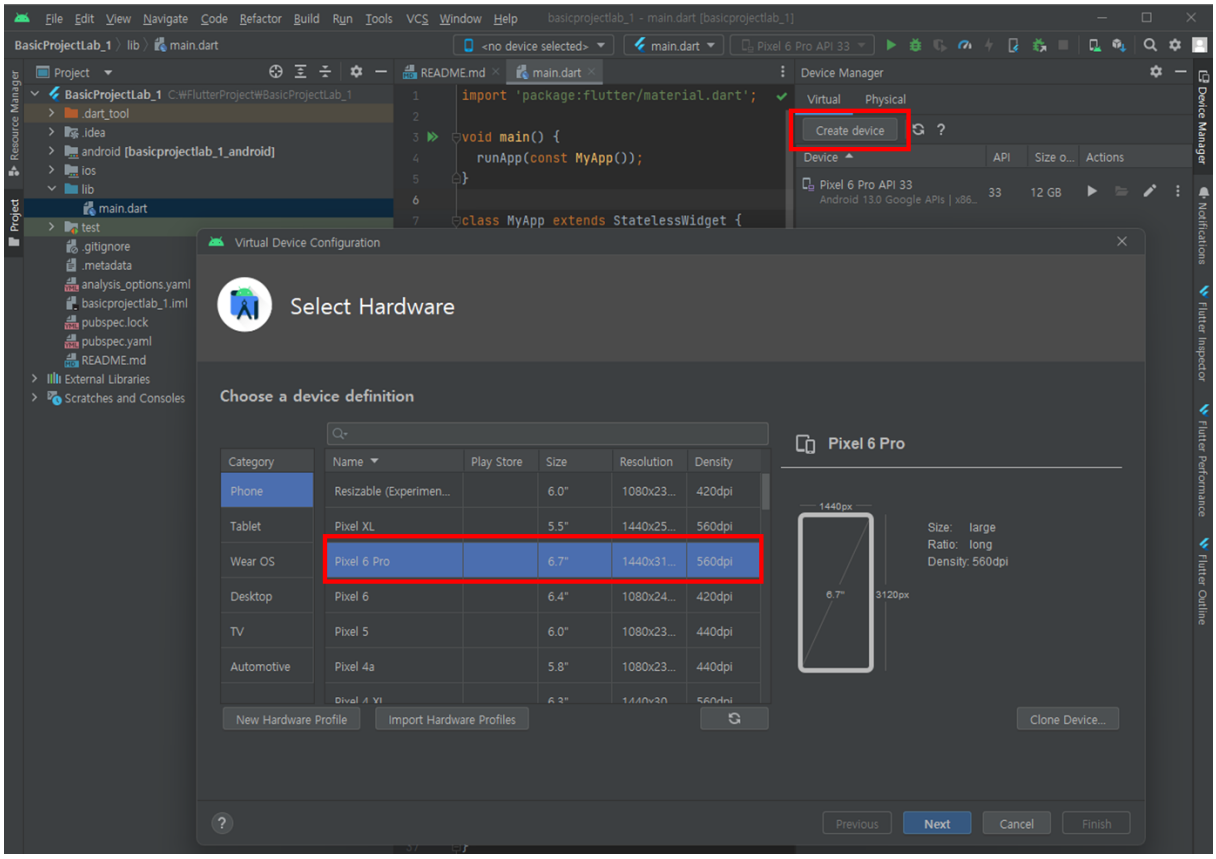

Settings
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
HTTP Proxy
Data Sharing
Date Formats
Updates
Process Elevation
Passwords
Android SDK
Memory Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Appearance & Behavior › System Settings › Android SDK
Manager for the Android SDK and Tools used by the IDE
Android SDK Location: C:₩Users₩picbu₩AppData₩Local₩Android₩Sdk
Edit
Optimize disk space
SDK Platforms
SDK Tools
SDK Update Sites
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.
Name
API Level
Revision
Status
Android UpsideDownCake Preview
UpsideDownCake
1
Not installed
TiramisuPrivacySandbox
9
Not installed
Android 13.0 (Tiramisu)
33
2
Installed
Android 12L (Sv2)
32
1
Not installed
Android 12.0 (S)
31
1
Partially installed
Android 11.0 (R)
30
3
Not installed
Android 10.0 (Q)
29
5
Not installed
Android 9.0 (Pie)
28
6
Not installed
Android 8.1 (Oreo)
27
3
Not installed
Android 8.0 (Oreo)
26
2
Not installed
Android 7.1.1 (Nougat)
25
3
Not installed
Android 7.0 (Nougat)
24
2
Not installed
Android 6.0 (Marshmallow)
23
3
Not installed
Android 5.1 (Lollipop)
22
2
Not installed
Android 5.0 (Lollipop)
21
2
Not installed
Android 4.4W (KitKat Wear)
20
2
Not installed
Android 4.4 (KitKat)
19
4
Not installed
Android 4.3 (Jelly Bean)
18
3
Not installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply

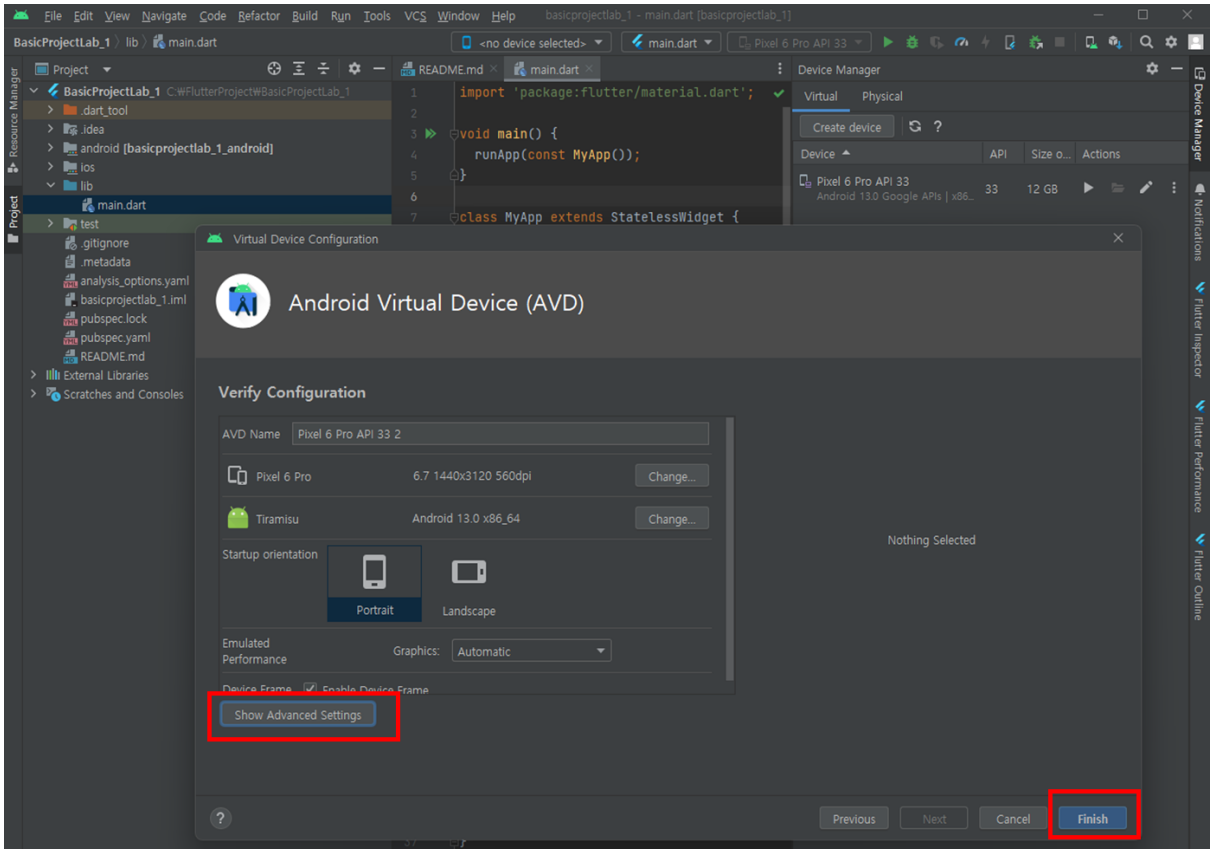

File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
New
Open...
Profile or Debug APK
Open Recent
Close Project
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Export
Print...
Power Save Mode
Exit
Resource Manager
Project
ProjectLab_1

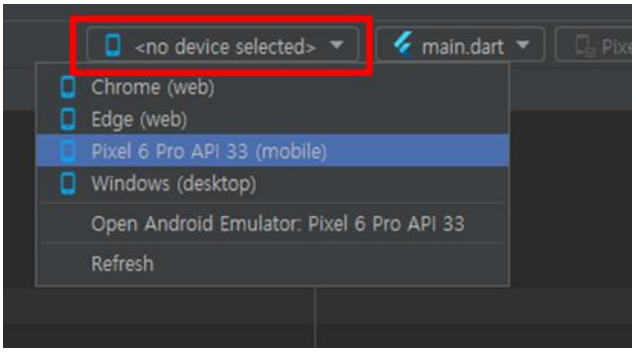

Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems 3
Project
Default settings for all modules. Configure these parameters for each module on the module page as needed.
Name: basicprojectlab_1
SDK: Android API 32 Platform version 11.0.15
Edit
Language level:
Compiler output:
<No SDK>
Android API 31 Platform version 11.0.15
Android API 32 Platform version 11.0.15
Android API 33, extension level 3 Platform version 11.0.15
11 version 11.0.15
jbr-11 JetBrains Runtime version 11.0.15
Add SDK
Detected SDKs
C:₩Program Files₩Java₩jdk-17.0.5 Oracle OpenJDK version 17.0.5
C:₩Program Files₩OpenJDK₩openjdk-11.0.15_10 version 11.0.15
C:₩Program Files₩Java₩jdk-11.0.14 Oracle OpenJDK version 11.0.14
ponding sources.

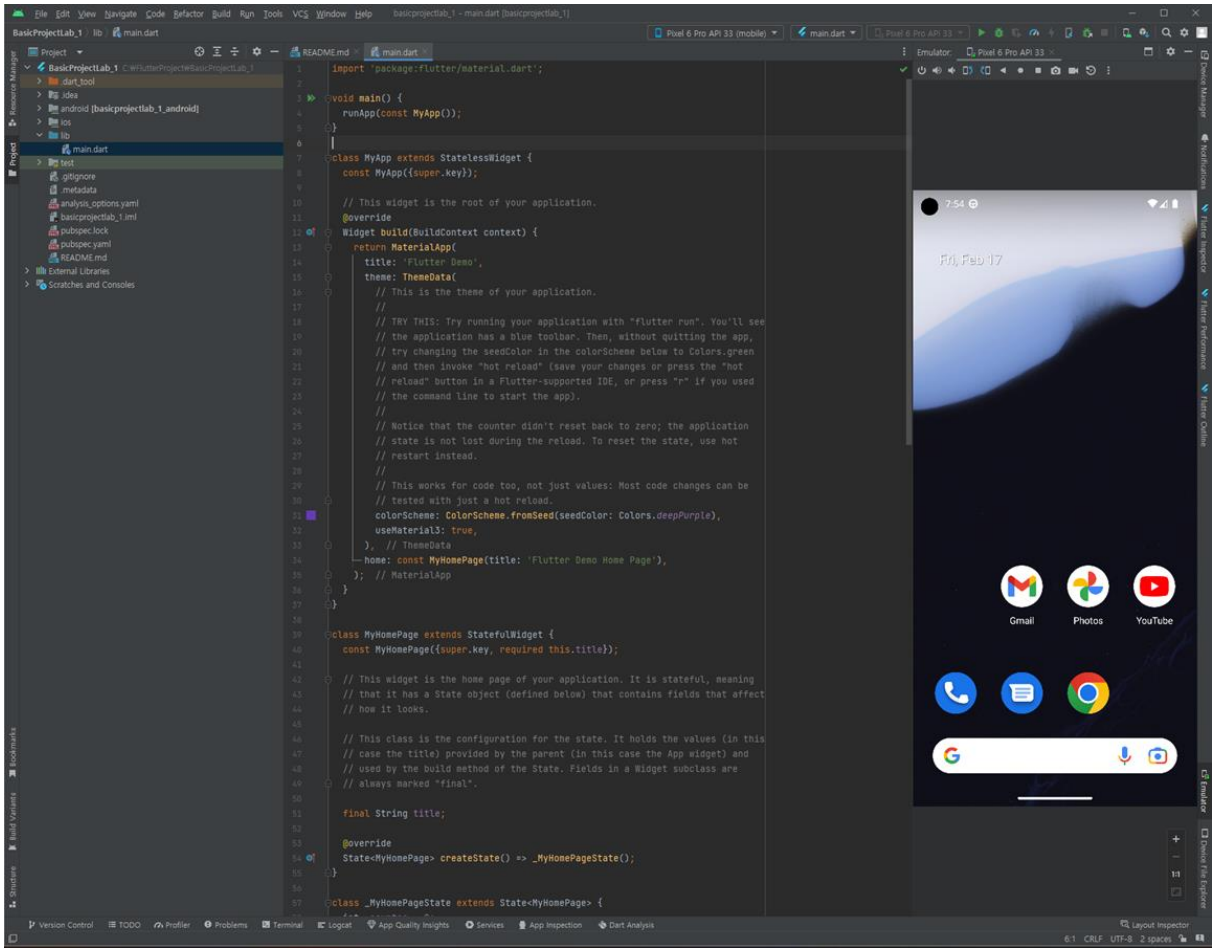

안드로이드 에뮬레이터 설치 필요
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
basicprojectlab_1 - main.dart [basicprojectlab_1]
BasicProjectLab_1 lib main.dart
<no device selected>
main.dart
Pixel 6 Pro API 33
Project
README.md
main.dart
BasicProjectLab_1 C:₩FlutterProject₩BasicProjectLab_1
.dart_tool
.idea
android [basicprojectlab_1_android]
ios
lib
main.dart
test
.gitignore
.metadata
analysis_options.yaml
basicprojectlab_1.iml
pubspec.lock
pubspec.yaml
README.md
External Libraries
Scratches and Consoles
import 'package:flutter/material.dart';
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({super.key});
// This widget is the root of your application.
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Flutter Demo',
theme: ThemeData(
// This is the theme of your application.
//
// TRY THIS: Try running your application with "flutter run". You'll see
// the application has a blue toolbar. Then, without quitting the app,
// try changing the seedColor in the colorScheme below to Colors.green
// and then invoke "hot reload" (save your changes or press the "hot
// reload" button in a Flutter-supported IDE, or press "r" if you used
// the command line to start the app).
//
// Notice that the counter didn't reset back to zero; the application
// state is not lost during the reload. To reset the state, use hot
// restart instead.
Device Manager
Notifications
Flutter Inspector
Flutter Performance
Flutter Outline

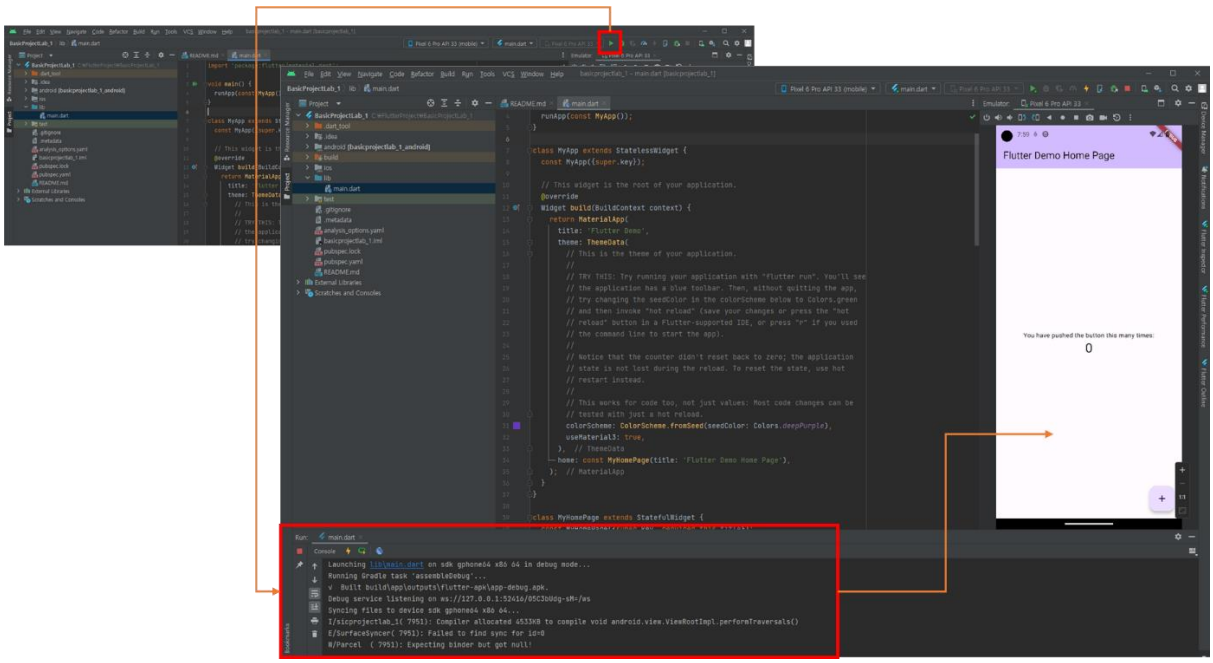

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
basicprojectlab_1 - main.dart [basicprojectlab_1]
BasicProjectLab_1 lib main.dart
<no device selected>
main.dart
Pixel 6 Pro API 33
Project
BasicProjectLab_1 C:\FlutterProject\BasicProjectLab_1
.dart_tool
.idea
android [basicprojectlab_1_android]
ios
lib
main.dart
test
.gitignore
.metadata
analysis_options.yaml
basicprojectlab_1.iml
pubspec.lock
pubspec.yaml
README.md
External Libraries
Scratches and Consoles
README.md
main.dart
import 'package:flutter/material.dart';
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
Device Manager
Virtual
Physical
Create device
Device
API
Size o...
Actions
Pixel 6 Pro API 33
Android 13.0 Google APIs | x86...
33
12 GB
Virtual Device Configuration
Select Hardware
Choose a device definition
Category
Phone
Tablet
Wear OS
Desktop
TV
Automotive
Name
Play Store
Size
Resolution
Density
Resizable (Experimen...
6.0"
1080x23...
420dpi
Pixel XL
5.5"
1440x25...
560dpi
Pixel 6 Pro
6.7"
1440x31...
560dpi
Pixel 6
6.4"
1080x24...
420dpi
Pixel 5
6.0"
1080x23...
440dpi
Pixel 4a
5.8"
1080x23...
440dpi
New Hardware Profile
Import Hardware Profiles
Pixel 6 Pro
1440px
6.7"
3120px
Size: large
Ratio: long
Density: 560dpi
Clone Device...
Previous
Next
Cancel
Finish
Device Manager
Notifications
Flutter Inspector
Flutter Performance
Flutter Outline

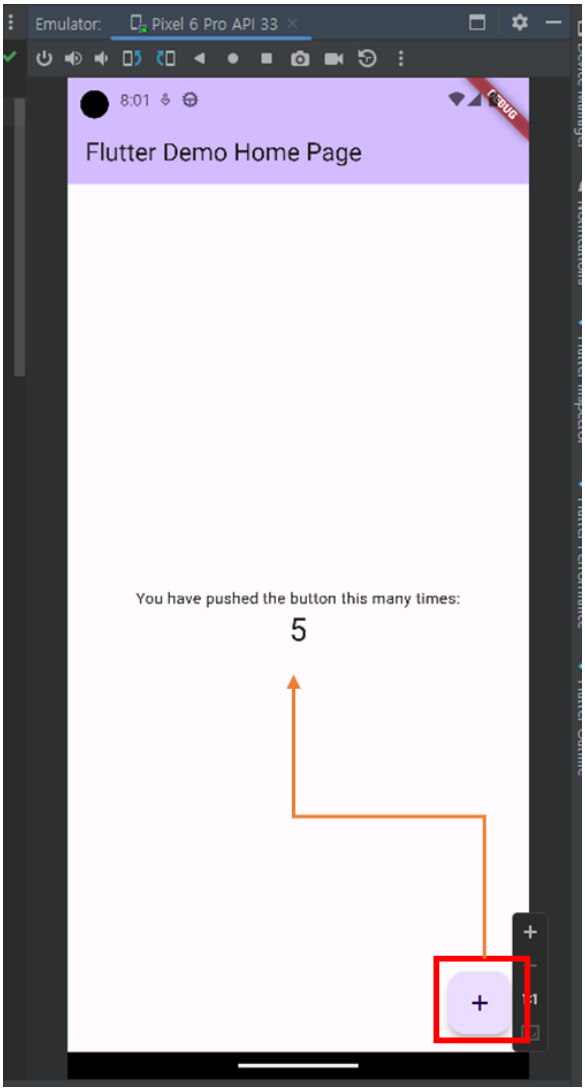

Virtual Device Configuration
Select Hardware
Choose a device definition
Category
Name
Play Store
Size
Resolution
Density
Phone
Tablet
Wear OS
Desktop
TV
Automotive
Resizable (Experimen...
Pixel XL
Pixel 6 Pro
Pixel 6
Pixel 5
Pixel 4a
6.0"
5.5"
6.7"
6.4"
6.0"
5.8"
1080x23...
1440x25...
1440x31...
1080x24...
1080x23...
1080x23...
420dpi
560dpi
560dpi
420dpi
440dpi
440dpi
Pixel 6 Pro
1440px
3120px
6.7"
Size: large
Ratio: long
Density: 560dpi
New Hardware Profile
Import Hardware Profiles
Clone Device...
Previous
Next
Cancel
Finish


Virtual Device Configuration
Android Virtual Device (AVD)
Verify Configuration
AVD Name
Pixel 6 Pro API 33 2
Pixel 6 Pro
6.7 1440x3120 560dpi
Change...
Tiramisu
Android 13.0 x86_64
Change...
Startup orientation
Portrait
Landscape
Emulated Performance
Graphics:
Automatic
Show Advanced Settings
Nothing Selected
Previous
Next
Cancel
Finish


<no device selected>
main.dart
Chrome (web)
Edge (web)
Pixel 6 Pro API 33 (mobile)
Windows (desktop)
Open Android Emulator: Pixel 6 Pro API 33
Refresh


import 'package:flutter/material.dart';
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({super.key});
// This widget is the root of your application.
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Flutter Demo',
theme: ThemeData(
// This is the theme of your application.
//
// TRY THIS: Try running your application with "flutter run". You'll see
// the application has a blue toolbar. Then, without quitting the app,
// try changing the seedColor in the colorScheme below to Colors.green
// and then invoke "hot reload" (save your changes or press the "hot
// reload" button in a Flutter-supported IDE, or press "r" if you used
// the command line to start the app).
//
// Notice that the counter didn't reset back to zero; the application
// state is not lost during the reload. To reset the state, use hot
// restart instead.
//
// This works for code too, not just values: Most code changes can be
// tested with just a hot reload.
colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
useMaterial3: true,
), // ThemeData
home: const MyHomePage(title: 'Flutter Demo Home Page'),
); // MaterialApp
}
}
class MyHomePage extends StatefulWidget {
const MyHomePage({super.key, required this.title});
// This widget is the home page of your application. It is stateful, meaning
// that it has a State object (defined below) that contains fields that affect
// how it looks.
// This class is the configuration for the state. It holds the values (in this
// case the title) provided by the parent (in this case the App widget) and
// used by the build method of the State. Fields in a Widget subclass are
// always marked "final".
final String title;
@override
State<MyHomePage> createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
Gmail
Photos
YouTube

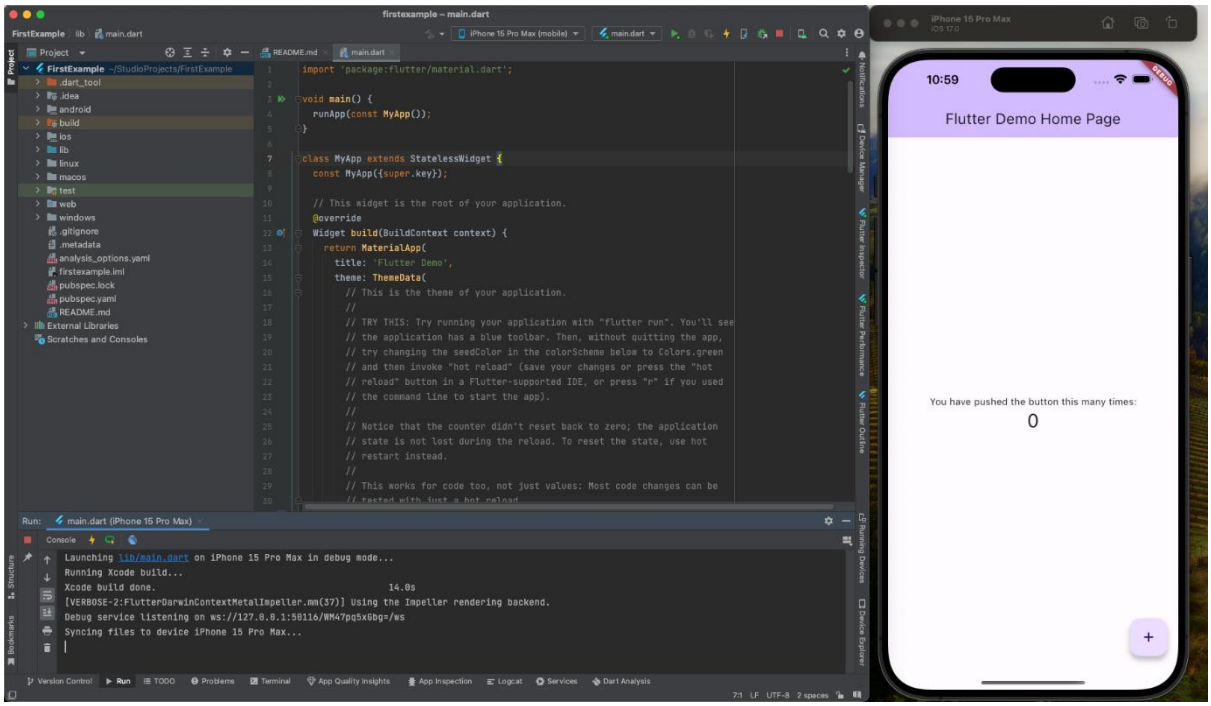

Flutter Demo Home Page
You have pushed the button this many times:
0

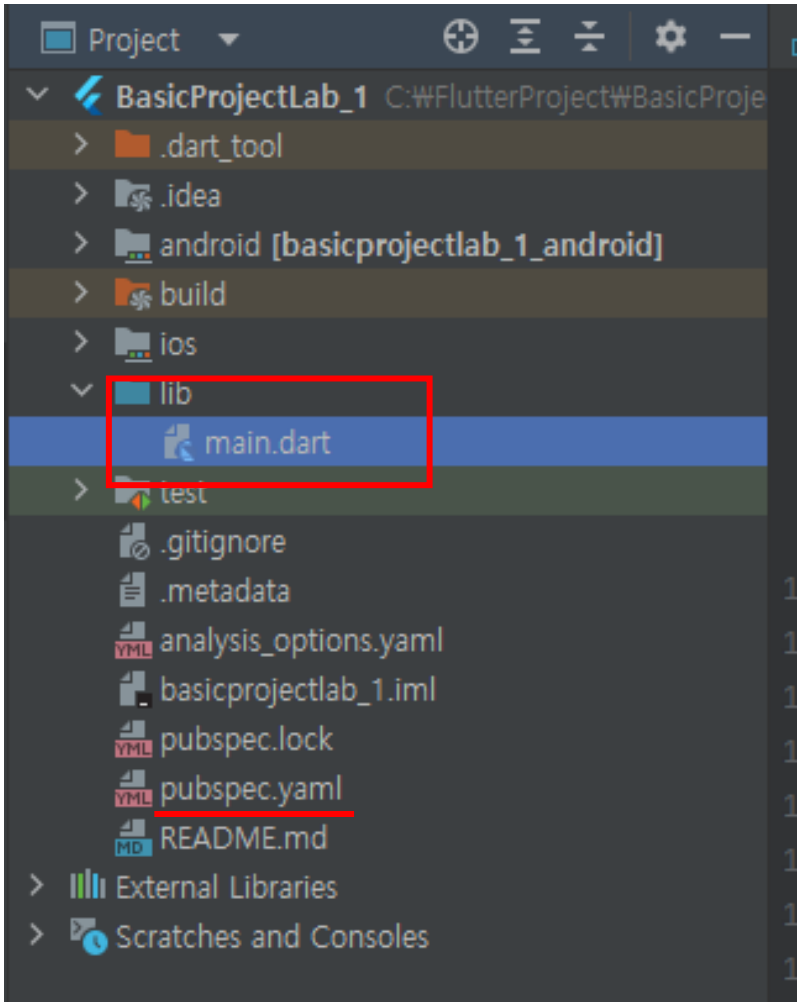

Emulator:
Pixel 6 Pro API 33
8:01
DEBUG
Flutter Demo Home Page
You have pushed the button this many times:
5


import 'package:flutter/material.dart';
void main() {
runApp(const MyApp());
}
class MyApp extends StatelessWidget {
const MyApp({super.key});
// This widget is the root of your application.
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Flutter Demo',
theme: ThemeData(
colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
useMaterial3: true,
), // ThemeData
home: const MyHomePage(title: 'Flutter Demo Home Page'),
); // MaterialApp
}
}
class MyHomePage extends StatefulWidget {
const MyHomePage({super.key, required this.title});
final String title;
@override
State<MyHomePage> createState() => _MyHomePageState();
}
Hot reload
Ctrl+₩
Ctrl+S

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {

      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {

    return Scaffold(
      appBar: AppBar(

        backgroundColor: Theme.of(context).colorScheme.inversePrimary,

        title: Text(widget.title),
      ),  // AppBar
      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),  // Text
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ),  // Text
          ],  // <Widget>[]
        ),  // Column
      ),  // Center
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.  // FloatingActionButton
    );  // Scaffold
  }
}
```

Mac





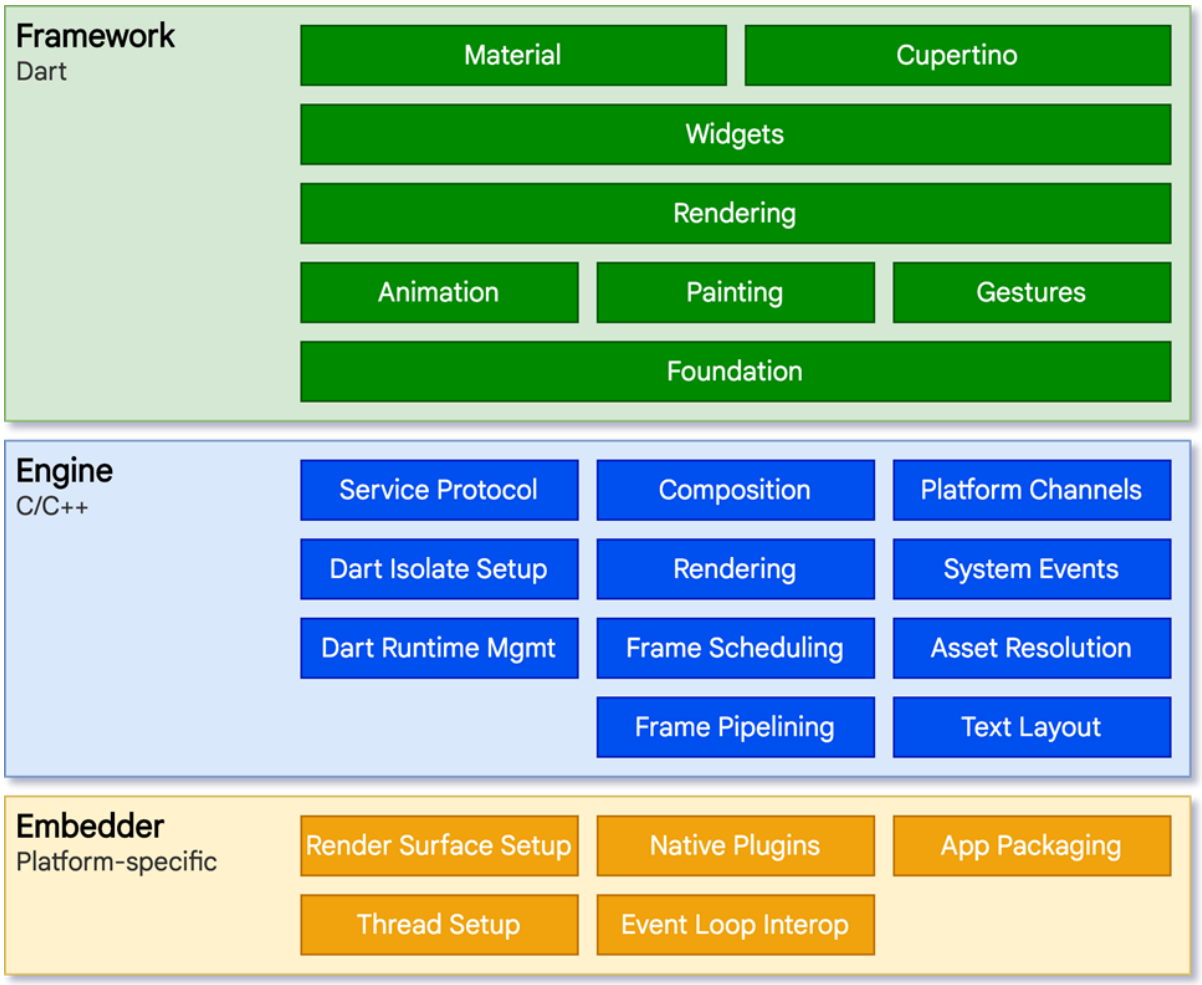
- .dart_tool : 다트의 여러 가지 도구를 관리
- .idea : 개발 도구에 필요한 설정
- android : 안드로이드 프로젝트 관련 파일→안드로이드 스튜디오로 실행 가능
- build : 빌드시 생성되는 파일
- ios : iOS 네이티브 코드를 작성하는 부분 ➔ 엑스코드로 실행 가능(맥 전용)
- lib : 플러터 앱 개발을 위한 다트 파일
- test : 테스트 코드를 작성하는 부분
- .gitignore : git 에 업로드할 때 필요 없는 파일 기록
- .metadata : 프로젝트가 관리하는 파일. 임의로 수정하면 안됨
- analysis_optioins.yaml : 다트 lint 설정 파일 (코드, 디버깅 룰)
- .packages : 각종 패키지 정보, 임의로 수정하면 안됨
- Flutter_app.iml : 개발도구에 필요한 설정 파일. 임의로 수정하면 안됨
- pubspec.lock : 패키지 매니저가 이용하는 파일. 임의로 수정하면 안됨
- pubspec.yaml : 패키지, 이미지, 폰트 설정
- README.md : 프로젝트 소개

- import 를 이용한 필요한 패키지 파일 호출
- 플러터는 main() 에서부터 시작
- runApp()을 통해 앱의 시작하는 함수 호출
- extends 는 상속을 의미
- Stateless Widget 이라는 클래스를 상속받음
- @override 애너테이션을 이용해서 build()라는 함수를 재정의
- runApp()을 이용해 클래스를 실행할 때는 MaterialApp() 함수를 반환 해야함
- title 은 앱의 이름을 정의
- theme 는 앱의 색이나 설정을 정의
- home 에는 앱을 실행할 때 첫 화면에 어떤 내용을 표시할지 정의

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),  // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );  // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
```

5-4. Flutter Architecture

https://docs.flutter.dev/resources/architectural-overview



5-5. Hello World

아래와 같이 전부 지우고 runApp 함수 확인해 보자

```dart
import 'package:flutter/material.dart';

void main(){
  runApp(const MaterialApp());
}
```

```
runApp(const MaterialApp());

package:flutter/src/widgets/binding.dart
void runApp(Widget app)

Type: void Function(Widget)
```

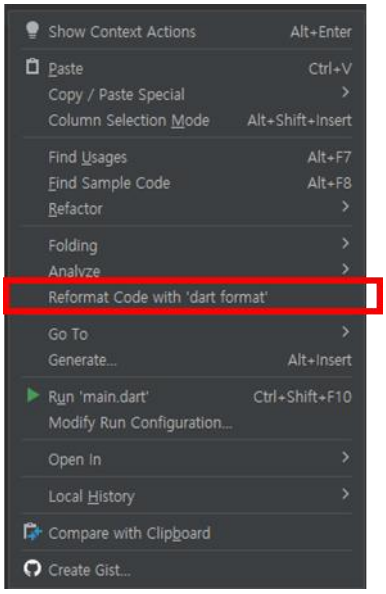
MaterialApp : 최상위 위젯

home: 필요!

Text 위젯으로 Hello World 출력

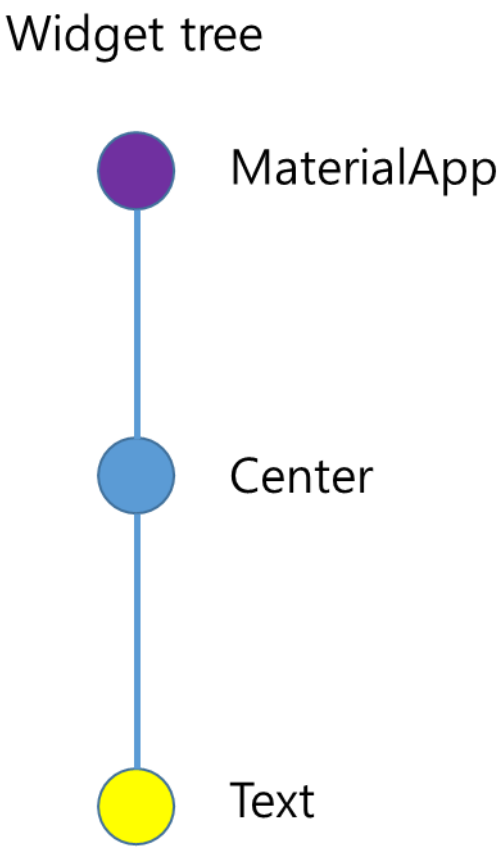
Text 위젯으로 Hello World 출력 → Center 위젯으로 중앙 정렬해 보자

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const MaterialApp(home: Center(
    child: Text('Hello World'),
  ),),);  // Center, MaterialApp
}
```



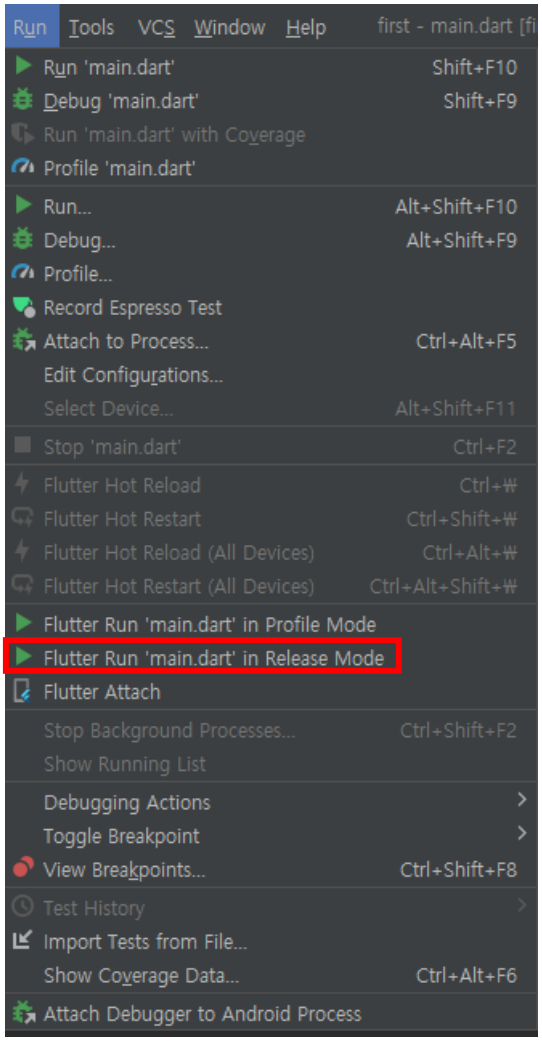
[실습]

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      home: Center(
        child: Text('Hello World'),
      ),  // Center
    ),  // MaterialApp
  );
}
```

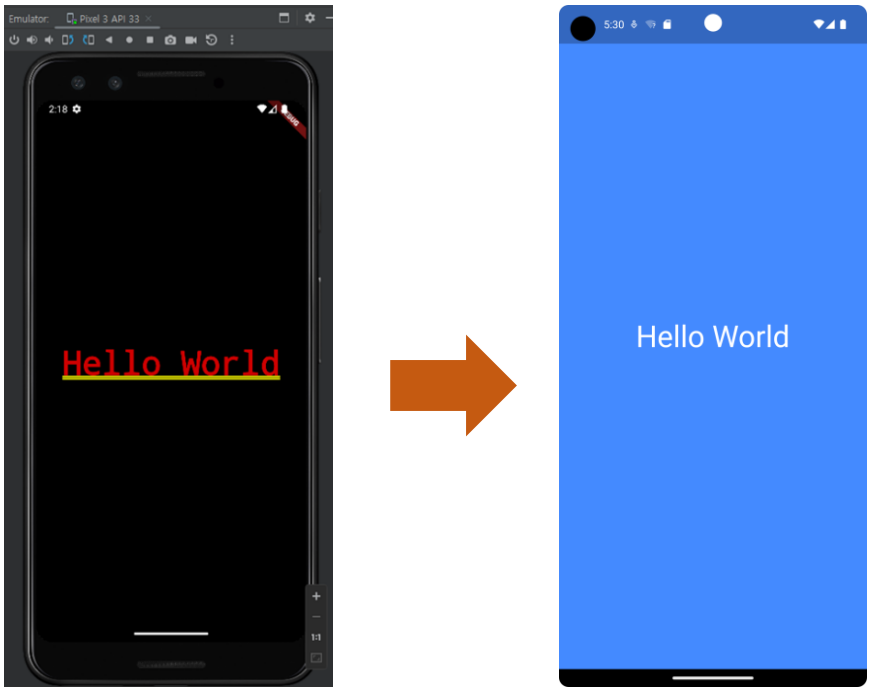


DEBUG" 배너 없애기

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Center(
        child: Text('Hello World'),
      ),  // Center
    ),  // MaterialApp
  );
}
```



"Hello World"가 너무 무서워요!!



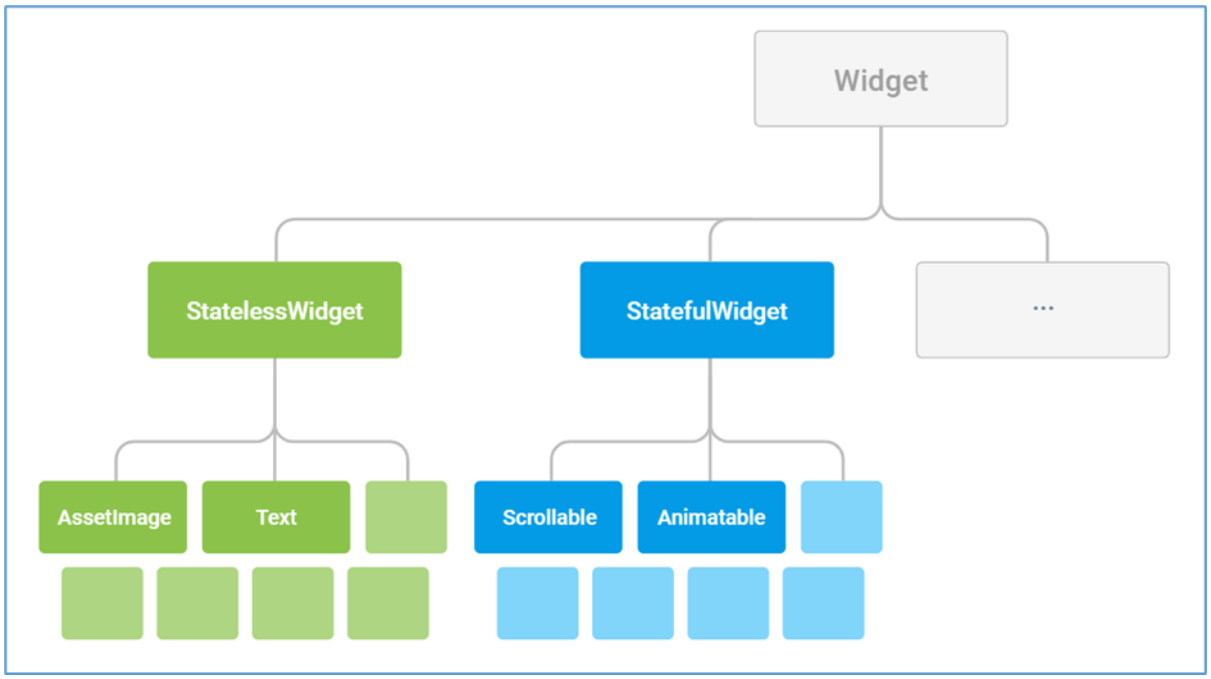
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Material(
        color: Colors.blue,
        child: Center(
          child : Text(
            'Hello World',
            style: TextStyle(fontSize: 40, color: Colors.white),
          ),  // Text
        ),  // Center
      ),  // Material
    ),  // MaterialApp
  );
}
```

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        backgroundColor: Colors.blueAccent,
        body: Center(
          child: Text(
            'Hello World',
            style: TextStyle(fontSize: 40, color: Colors.white),
          ),  // Text
        ),  // Center
      ),  // Scaffold
    ),  // MaterialApp
  );
}
```

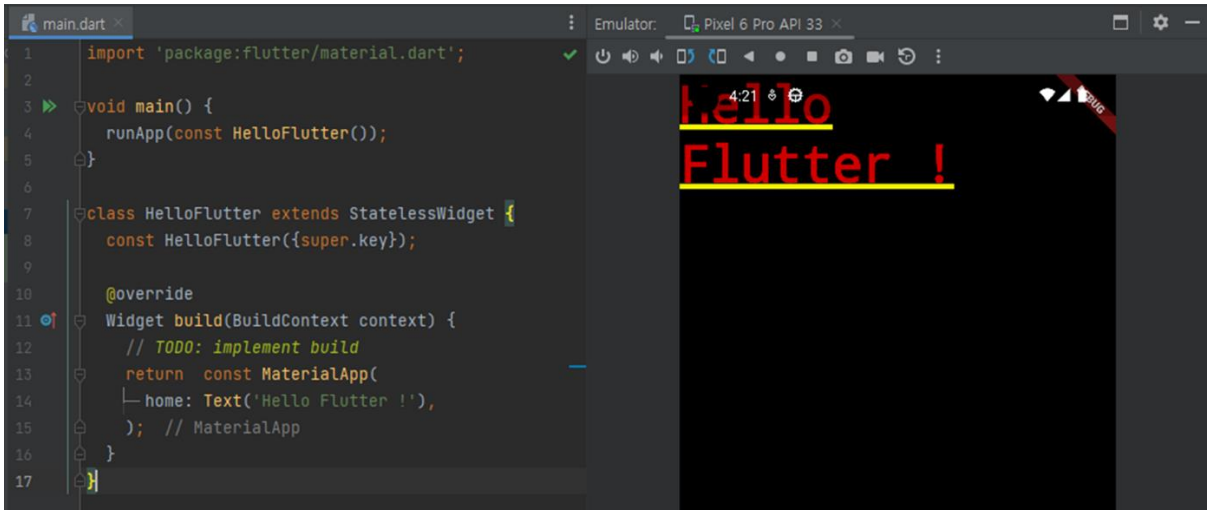
5-6. StatefulWidget vs StatelessWidget

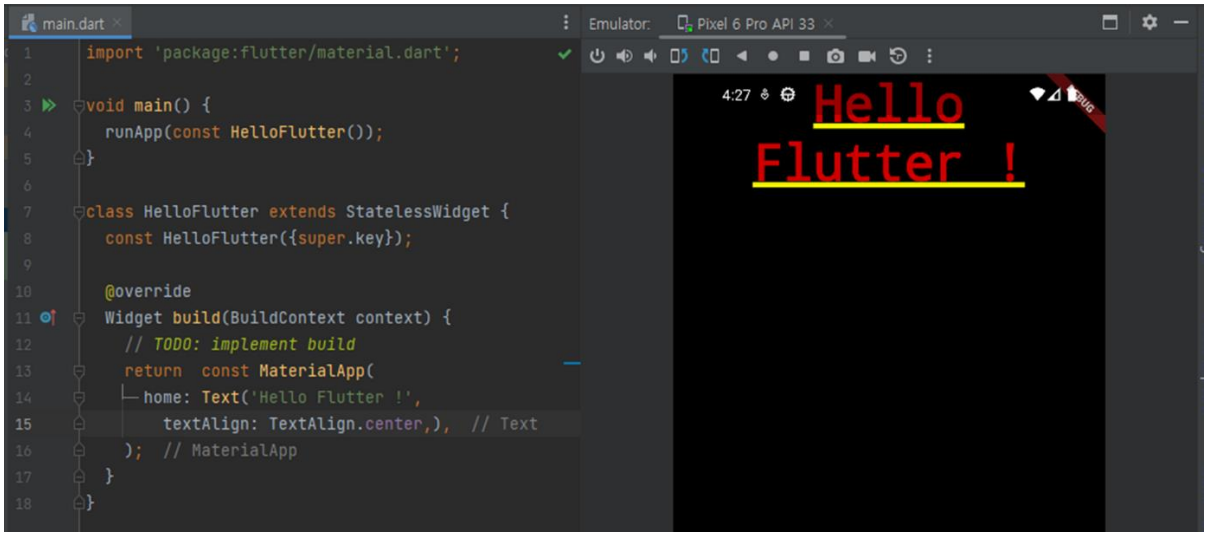
모든 것은 위젯입니다

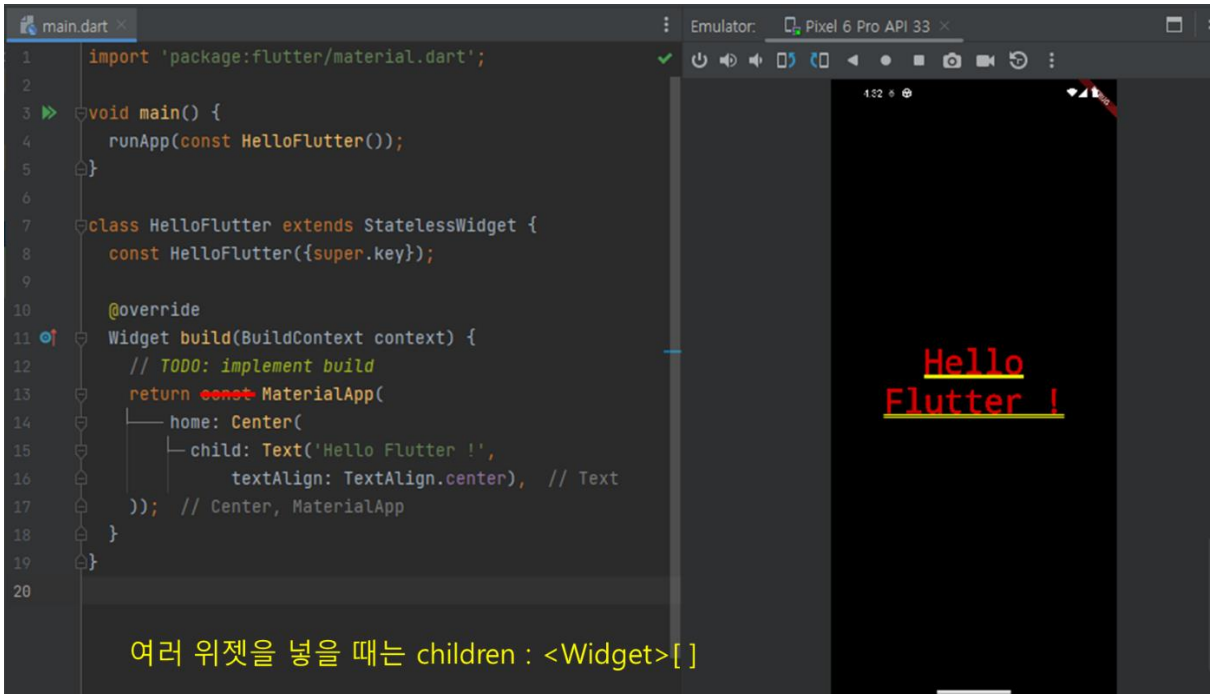
위젯은 Flutter 앱 UI 의 기본 단위입니다.



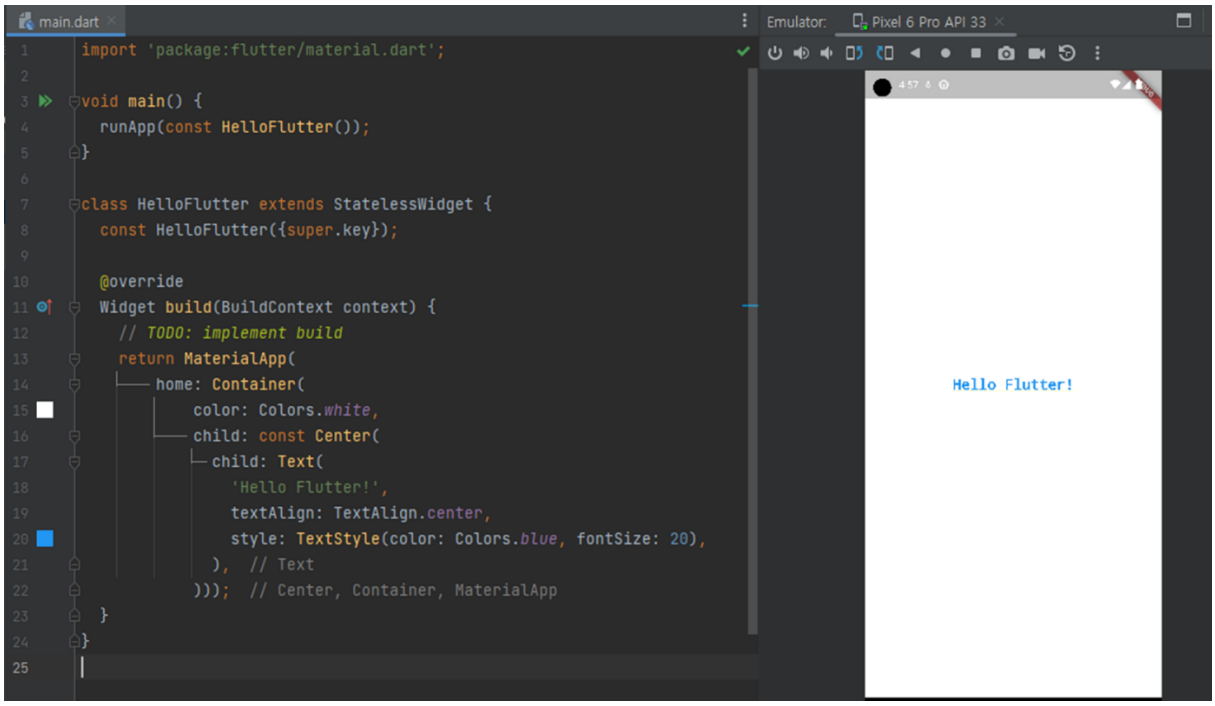
만약 위젯 고유의 특성이 사용자 상호작용이나 기타 다른 요소에 의해 변경될 필요가 있다면 위젯은 *stateful* 이라고 말할 수 있습니다. 예를 들어, 만약 사용자가 버튼을 누를 때마다 1 씩 증가하는 카운터 위젯이 있다고 하면, 그 카운터의 값은 위젯의 상태입니다. 값이 변할 때마다, 위젯은 UI 를 업데이트하기 위해 다시 빌드되어야 합니다.
이러한 위젯은 StatefulWidget(StatelessWidget 과는 다른)의 서브 클래스이며 변경 가능한 상태를 서브 클래스의 State 에 저장합니다.


main.dart
import 'package:flutter/material.dart';
void main() {
runApp(const HelloFlutter());
}
class HelloFlutter extends StatelessWidget {
const HelloFlutter({super.key});
@override
Widget build(BuildContext context) {
// TODO: implement build
return const MaterialApp(
home: Text('Hello Flutter !'),
); // MaterialApp
}
}
Emulator: Pixel 6 Pro API 33
Hello
Flutter !


main.dart
import 'package:flutter/material.dart';
void main() {
runApp(const HelloFlutter());
}
class HelloFlutter extends StatelessWidget {
const HelloFlutter({super.key});
@override
Widget build(BuildContext context) {
// TODO: implement build
return const MaterialApp(
home: Text('Hello Flutter !',
textAlign: TextAlign.center,), // Text
); // MaterialApp
}
}
Emulator: Pixel 6 Pro API 33
Hello
Flutter !


main.dart
import 'package:flutter/material.dart';
void main() {
runApp(const HelloFlutter());
}
class HelloFlutter extends StatelessWidget {
const HelloFlutter({super.key});
@override
Widget build(BuildContext context) {
// TODO: implement build
return const MaterialApp(
home: Center(
child: Text('Hello Flutter !',
textAlign: TextAlign.center), // Text
)); // Center, MaterialApp
}
}
Emulator: Pixel 6 Pro API 33
Hello
Flutter !
여러 위젯을 넣을 때는 children : <Widget>[ ]

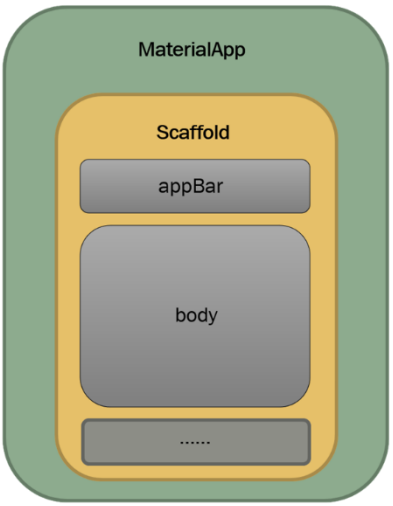
[실습]




```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const HelloFlutter());
}

class HelloFlutter extends StatelessWidget {
  const HelloFlutter({super.key});

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return MaterialApp(
      home: Container(
        color: Colors.white,
        child: const Center(
          child: Text(
            'Hello Flutter!',
            textAlign: TextAlign.center,
            style: TextStyle(color: Colors.blue, fontSize: 20),
          ),  // Text
        ));  // Center, Container, MaterialApp
  }
}
```

MaterialApp - Scaffold

MaterialApp	Scaffold
하위페이지나 구성요소를 담는 최상위 위젯	페이지의 형태
• title • theme • home	• appbar • body



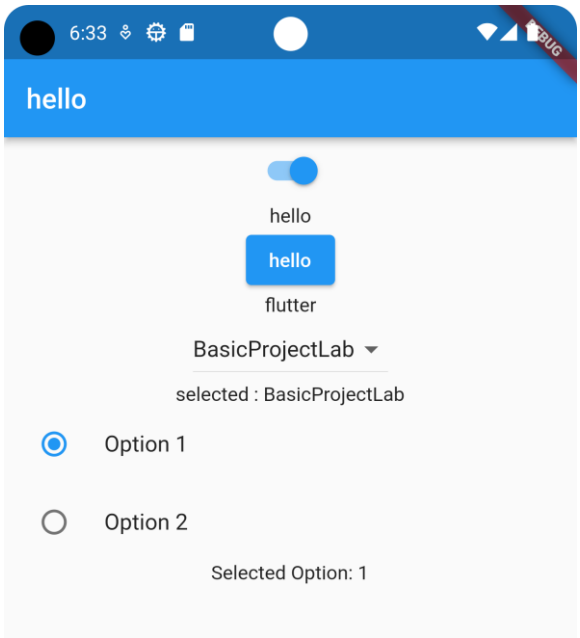
scaffolding

Stateful Widget – Material style

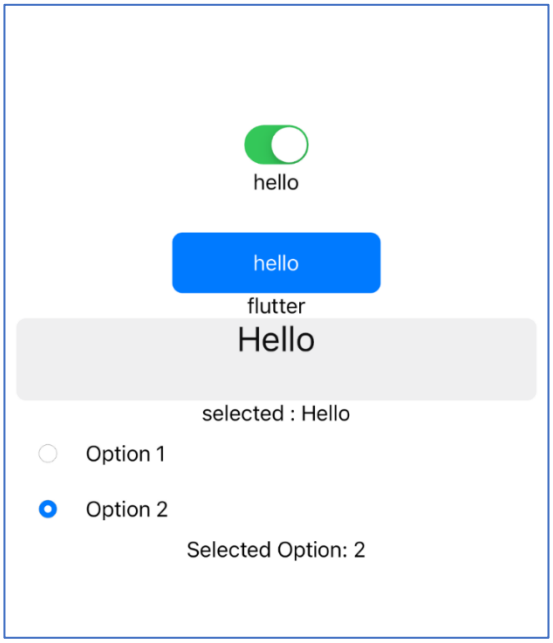
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const Hello());
}

class Hello extends StatefulWidget {
  const Hello({super.key});

  @override
  State<StatefulWidget> createState() => _Hello();
}

class _Hello extends State<Hello> {
  var switchValue = false;
  String strText = '';
  String strButton = '';
  List<String> drop = ['Hello', 'Flutter', 'BasicProjectLab'];
  String selectedDrop = 'Hello';
  int? selectedOption;

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return MaterialApp(
      title: 'Hello',
      theme: ThemeData(),
      home: Scaffold(
          appBar: AppBar(
            title: const Text('hello'),
          ),
          body: Column(
            children: <Widget>[
              Switch(
                value: switchValue,
                onChanged: (bool value) {
                  setState(() {
                    switchValue = value;
                    strText = switchValue ? 'hello' : '';
                  });
                },
              ),
              Text(strText),
              ElevatedButton(
                  onPressed: () {
                    setState(() {
                      strButton = (strText == 'hello') ? 'flutter' : '';
                    });
                  },
                  child: Text(strText)),
              Text(strButton),
              DropdownButton(
                value: selectedDrop,
                items: drop.map((String item) {
                  return DropdownMenuItem<String>(
                    value: item,
                    child: Text(item),
```

```
                    );
                  }).toList(),
                  onChanged: (dynamic value) {
                    setState(() {
                      selectedDrop = value;
                    });
                  },),
                Text('selected : $selectedDrop'),
                RadioListTile<int>(
                  title: const Text('Option 1'),
                  value: 1,
                  groupValue: selectedOption,
                  onChanged: (int? value) {
                    setState(() {
                      selectedOption = value;
                    });
                  },
                ),
                RadioListTile<int>(
                  title: const Text('Option 2'),
                  value: 2,
                  groupValue: selectedOption,
                  onChanged: (int? value) {
                    setState(() {
                      selectedOption = value;
                    });
                  },
                ),
                Text("Selected Option: $selectedOption"),
              ],
            )),
      );
  }
}
```

Material Style



Cupertino style



Stateful Widget – Cupertino style

```
import 'package:flutter/cupertino.dart';

void main() {
 runApp(const Hello());
}

class Hello extends StatefulWidget {
 const Hello({super.key});

 @override
 State<StatefulWidget> createState() => _Hello();
}

class _Hello extends State<Hello> {
 var switchValue = false;
 String strText = '';
 String strButton = '';
 List<String> drop = ['Hello', 'Flutter', 'BasicProjectLab'];
 String selectedDrop = 'Hello';
 int? selectedOption;

 @override
 Widget build(BuildContext context) {
   // TODO: implement build
   return CupertinoApp(
     title: 'Hello',
     theme: const CupertinoThemeData(),
     home: CupertinoPageScaffold(
       child: Center(
         child: Column(
           mainAxisSize: MainAxisSize.min,
           children: <Widget>[
             CupertinoSwitch(
               value: switchValue,
               onChanged: (bool value) {
                 setState(() {
                   switchValue = value;
                   strText = switchValue ? 'hello' : '';
                 });
               },
             ),
             Text(strText),
             const SizedBox(height: 30),
             CupertinoButton.filled(
                 onPressed: () {
                   setState(() {
            strButton = (strText == 'hello') ? 'flutter' : '';
                   });
                 },
```

```
                child: Text(strText)),
            Text(strButton),
            CupertinoPicker(
              onSelectedItemChanged: (value) {
                selectedDrop = drop[value];
                setState(() {});
              },
              itemExtent: 50,
              diameterRatio: 1,
              useMagnifier: true,
              magnification: 1.3,
              scrollController: FixedExtentScrollController(
                initialItem: 0,
              ),
              children: [
                Text(drop[0]), Text(drop[1]), Text(drop[2]),
              ],
            ),
            Text('selected : $selectedDrop'),
            CupertinoListTile(
              title: const Text('Option 1'),
              leading: CupertinoRadio<int>(
                value: 1,
                groupValue: selectedOption,
                onChanged: (int? value) {
                  setState(() {
                    selectedOption = value;
                  });
                },
              ),
            ),
            CupertinoListTile(
              title: const Text('Option 2'),
              leading: CupertinoRadio<int>(
                value: 2,
                groupValue: selectedOption,
                onChanged: (int? value) {
                  setState(() {
                    selectedOption = value;
                  });
                },
              ),
            ),
            Text("Selected Option: $selectedOption"),
          ],
        ),
      ),
    ),
  );
 }
}
```

Stateful and stateless widgets

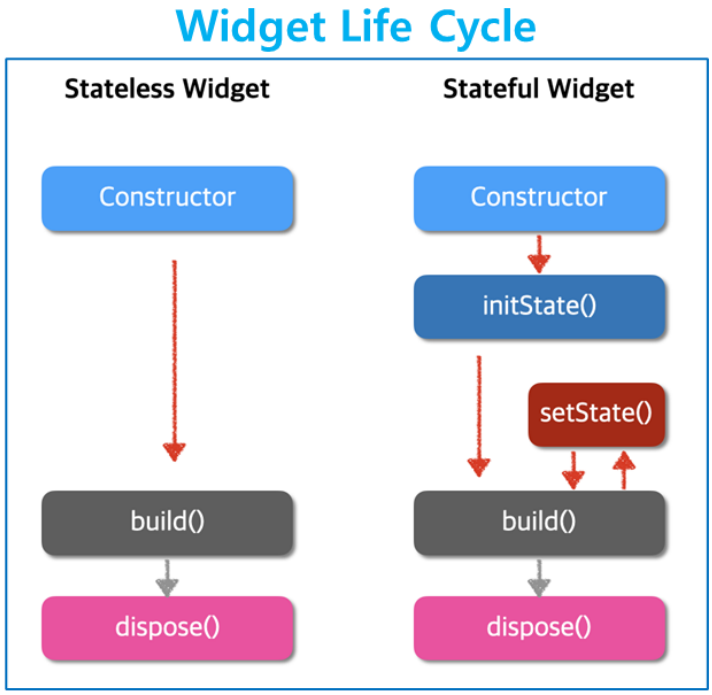
A widget is either stateful or stateless. If a widget can change—when a user interacts with it, for example—it's stateful.

A *stateless* widget never changes. Icon, IconButton, and Text are examples of stateless widgets. Stateless widgets subclass StatelessWidget.

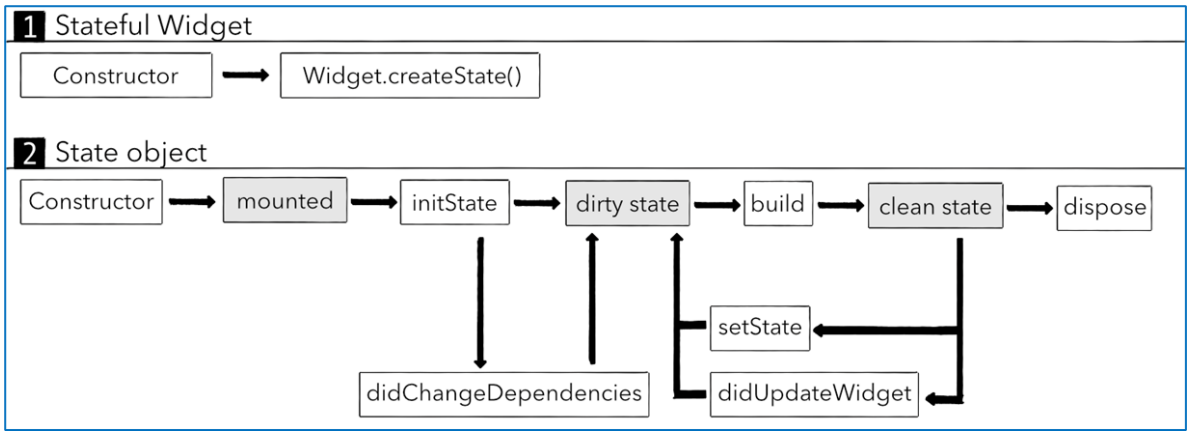
A *stateful* widget is dynamic: for example, it can change its appearance in response to events triggered by user interactions or when it receives data. Checkbox, Radio, Slider, InkWell, Form, and TextField are examples of stateful widgets. Stateful widgets subclass StatefulWidget.

A widget's state is stored in a State object, separating the widget's state from its appearance. The state consists of values that can change, like a slider's current value or whether a checkbox is checked. When the widget's state changes, the state object calls `setState()`, telling the framework to redraw the widget.

5-7. Life cycle

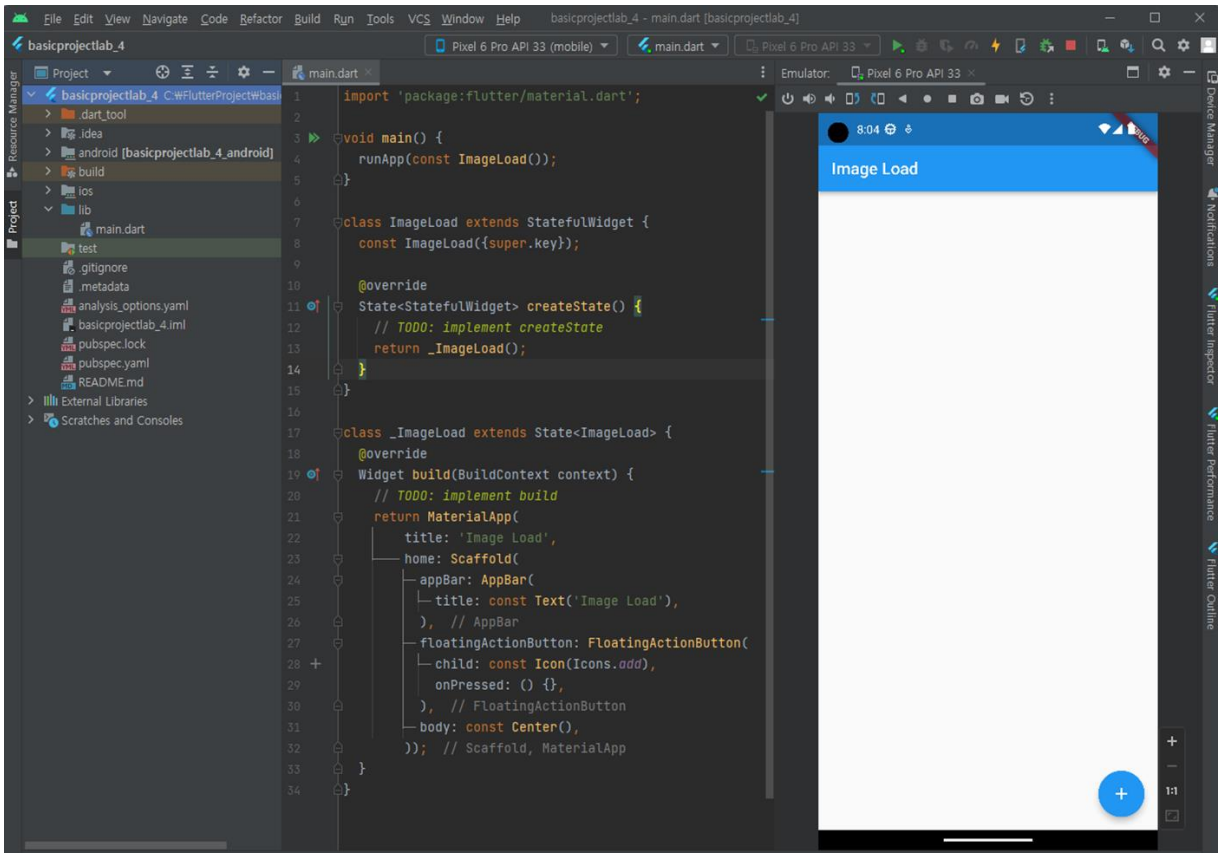


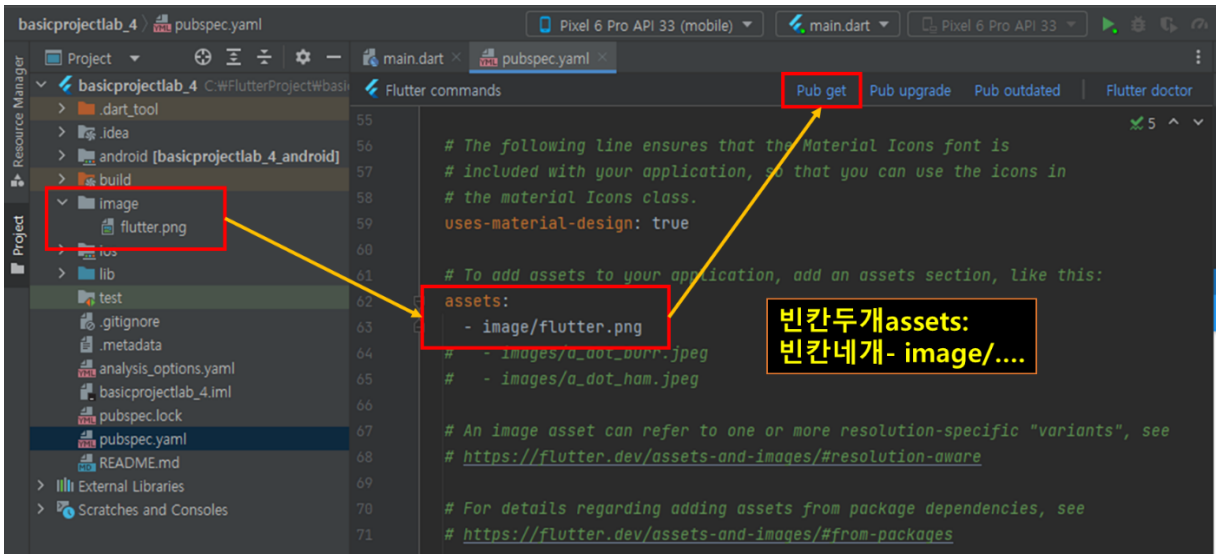
createState(), initState(), build(), setState(), dispose() → override method

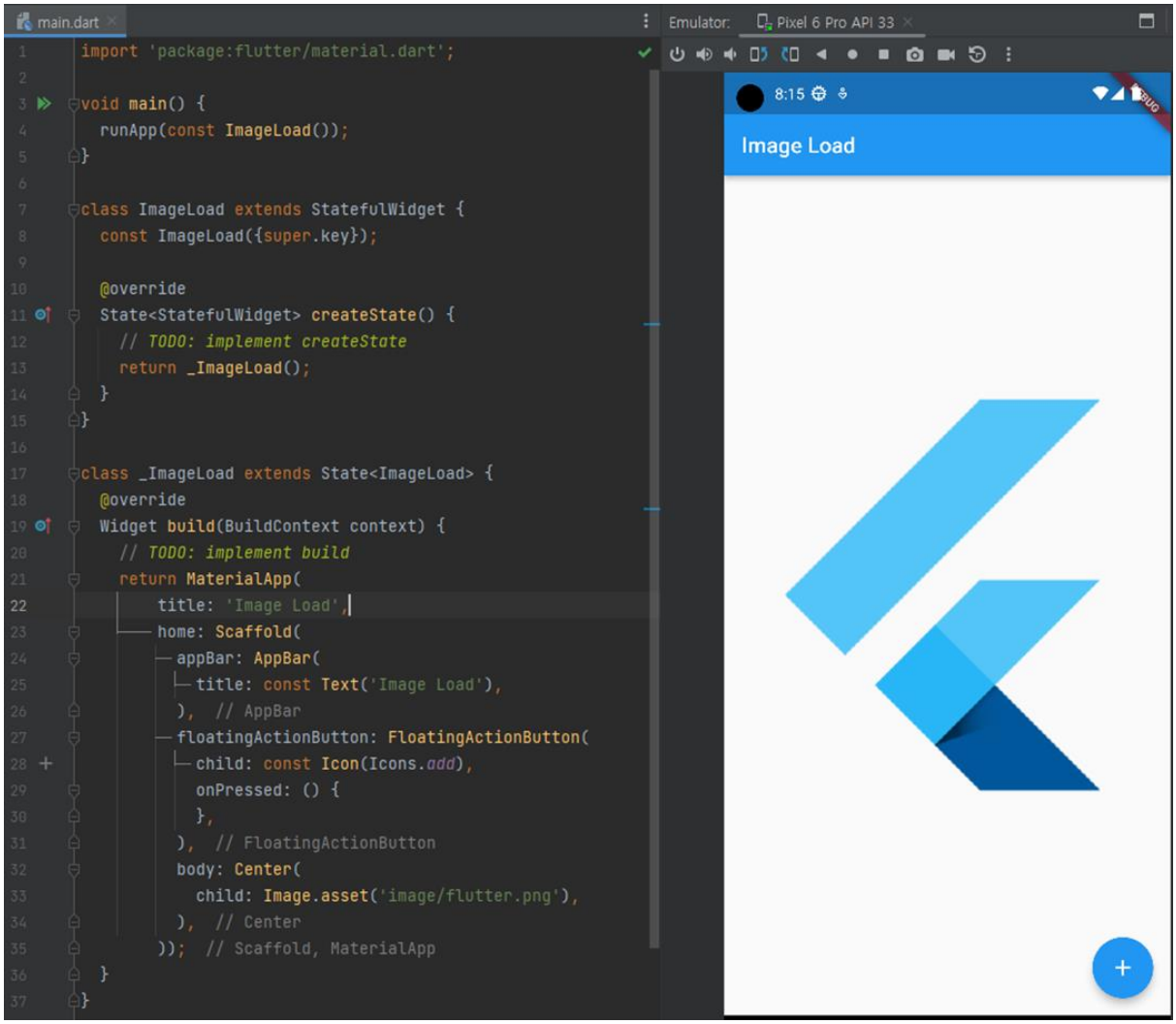


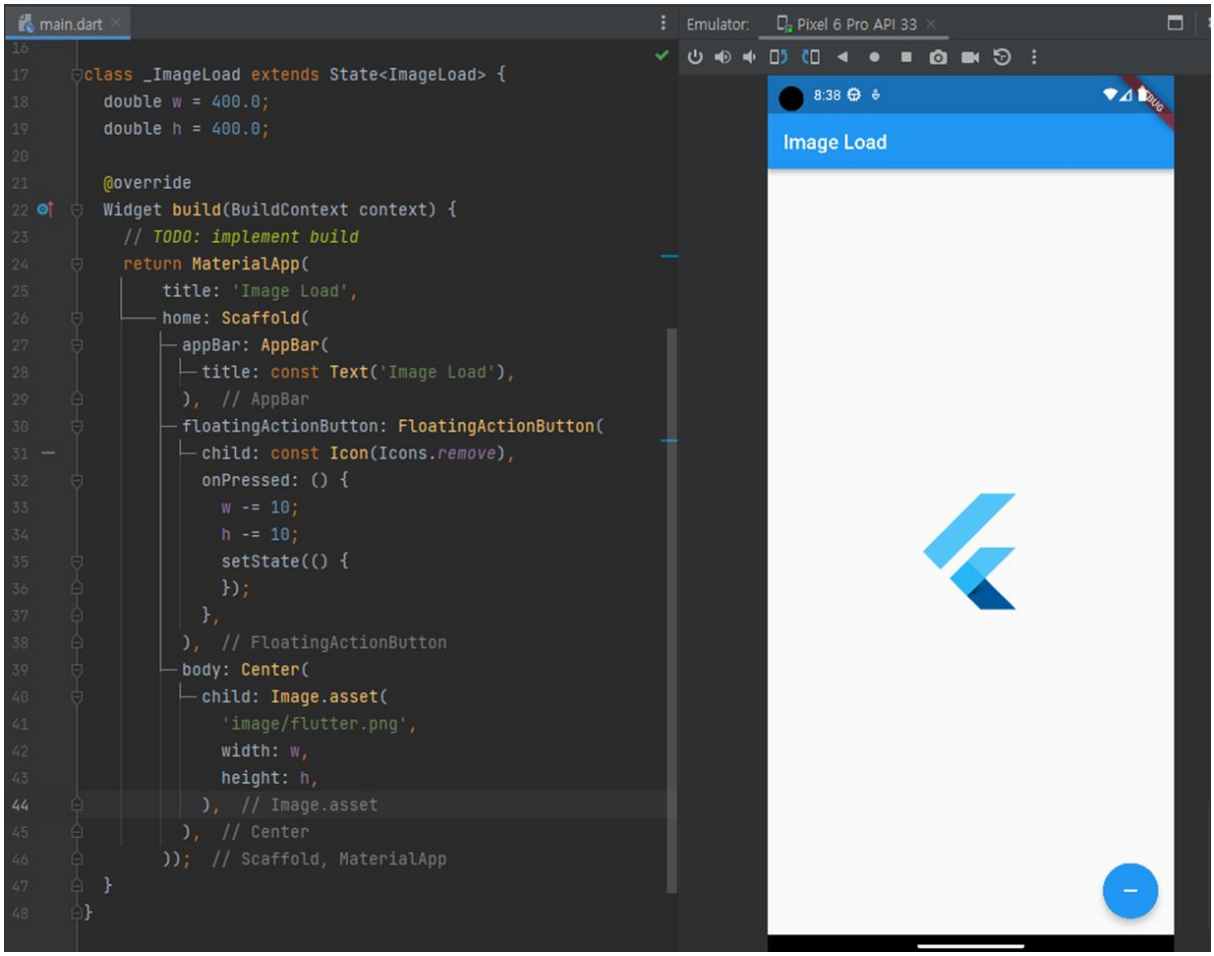
State Construction	Rendering	State Deconstruction
createState() statefulWidget 을 구축하자마자 호출됩니다. 위젯 트리에 상태를 만들기 위해 호출됩니다.	**build()** 위젯으로 만든 UI 를 구축합니다. 다양한 곳에서 반복적으로 호출됩니다. 변경된 부분 트리를 감지하고 대체합니다.	**deactivate()** state 오브젝트가 트리로부터 삭제될 때마다 호출됩니다.
initState() 위젯 트리 초기화를 합니다. 단 한 번만 호출됩니다.	**didUpdateWidget()** 위젯의 구성이 변경될 때마다 호출됩니다. 부모 위젯이 변경되고 다시 그려져야 할 때 호출됩니다. oldWidget 인수를 취득해 비교합니다.	**dispose()** 객체가 트리에서 완전히 삭제되고 두 번 다시 빌드되지 않으면 호출됩니다.
didChangeDependencies() state 객체의 종속성이 변경될 때 호출됩니다. initState 뒤에 호출되지만 그 이외에도 호출됩니다.	**setState()** 상태가 변경되었을 때 프레임워크에 상태가 변경됨을 알립니다.	

[실습]


basicprojectlab_4 - main.dart [basicprojectlab_4]
Pixel 6 Pro API 33 (mobile)
main.dart
Emulator: Pixel 6 Pro API 33
Image Load
import 'package:flutter/material.dart';
void main() {
runApp(const ImageLoad());
}
class ImageLoad extends StatefulWidget {
const ImageLoad({super.key});
@override
State<StatefulWidget> createState() {
// TODO: implement createState
return _ImageLoad();
}
}
class _ImageLoad extends State<ImageLoad> {
@override
Widget build(BuildContext context) {
// TODO: implement build
return MaterialApp(
title: 'Image Load',
home: Scaffold(
appBar: AppBar(
title: const Text('Image Load'),
), // AppBar
floatingActionButton: FloatingActionButton(
child: const Icon(Icons.add),
onPressed: () {},
), // FloatingActionButton
body: const Center(),
)); // Scaffold, MaterialApp
}
}

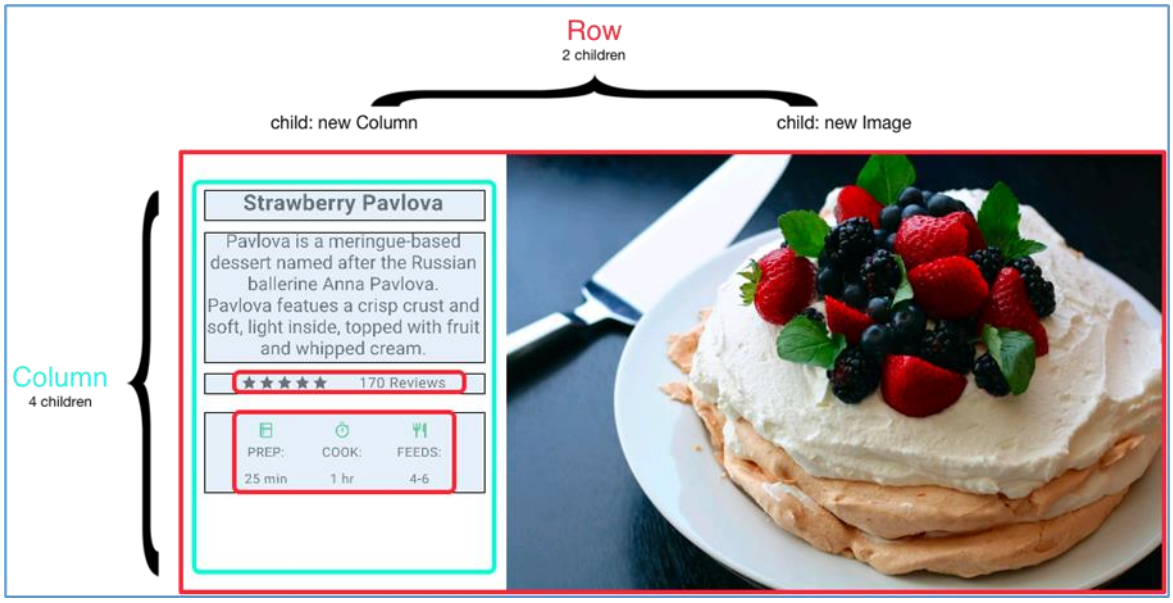

basicprojectlab_4 pubspec.yaml
Pixel 6 Pro API 33 (mobile)
main.dart
Pixel 6 Pro API 33
Flutter commands
Pub get
Pub upgrade
Pub outdated
Flutter doctor
image
flutter.png
assets:
- image/flutter.png
빈칸두개assets:
빈칸네개- image/....
The following line ensures that the Material Icons font is
included with your application, so that you can use the icons in
the material Icons class.
uses-material-design: true
To add assets to your application, add an assets section, like this:
An image asset can refer to one or more resolution-specific "variants", see
https://flutter.dev/assets-and-images/#resolution-aware
For details regarding adding assets from package dependencies, see
https://flutter.dev/assets-and-images/#from-packages

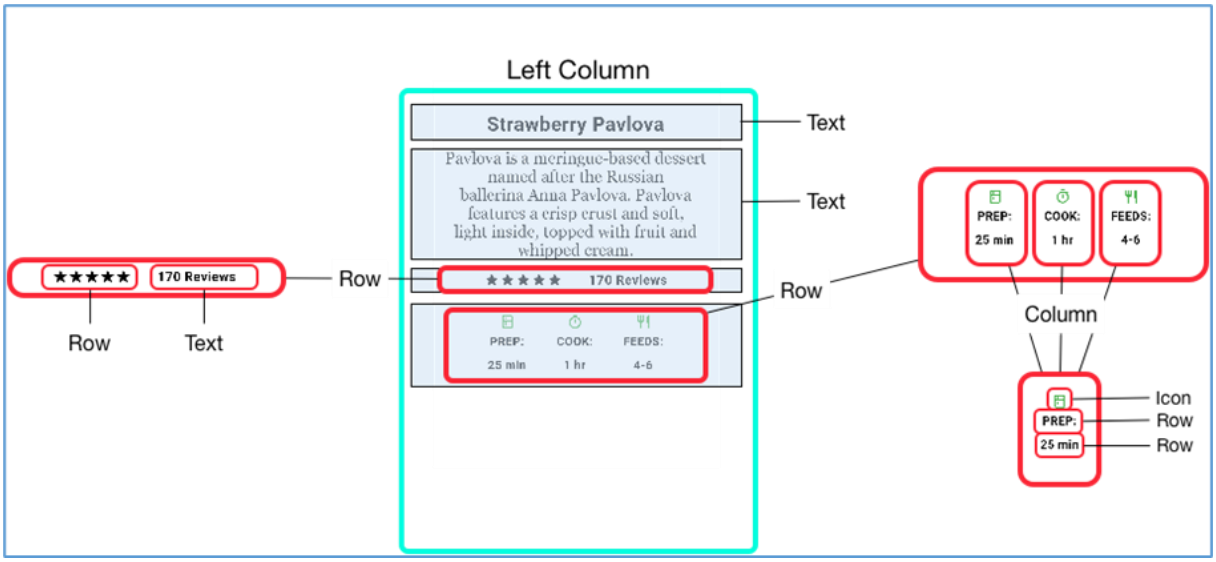

main.dart
import 'package:flutter/material.dart';
void main() {
runApp(const ImageLoad());
}
class ImageLoad extends StatefulWidget {
const ImageLoad({super.key});
@override
State<StatefulWidget> createState() {
// TODO: implement createState
return _ImageLoad();
}
}
class _ImageLoad extends State<ImageLoad> {
@override
Widget build(BuildContext context) {
// TODO: implement build
return MaterialApp(
title: 'Image Load',
home: Scaffold(
appBar: AppBar(
title: const Text('Image Load'),
), // AppBar
floatingActionButton: FloatingActionButton(
child: const Icon(Icons.add),
onPressed: () {
},
), // FloatingActionButton
body: Center(
child: Image.asset('image/flutter.png'),
), // Center
)); // Scaffold, MaterialApp
}
}
Emulator:
Pixel 6 Pro API 33
8:15
Image Load



UI Layout

https://docs.flutter.dev/ui/layout



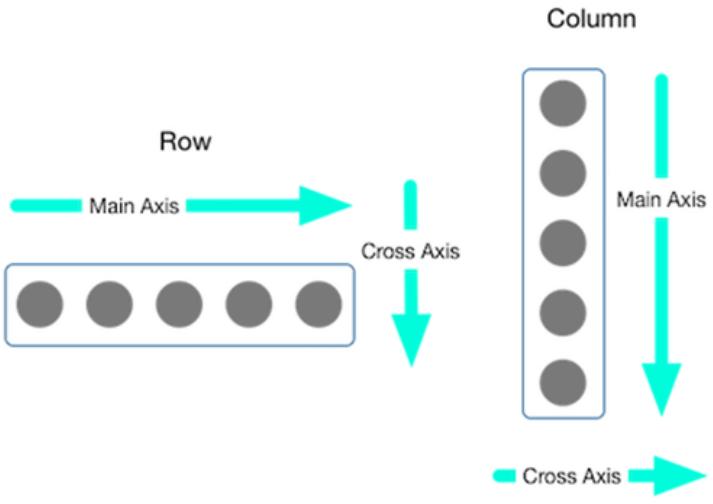


All layout widgets have either of the following:

- A `child` property if they take a single child—for example, `Center` or `Container`
- A `children` property if they take a list of widgets—for example, `Row`, `Column`, `ListView`, or `Stack`.

Aligning widgets

You control how a row or column aligns its children using the `mainAxisAlignment` and `crossAxisAlignment` properties. For a row, the main axis runs horizontally and the cross axis runs vertically. For a column, the main axis runs vertically and the cross axis runs horizontally.

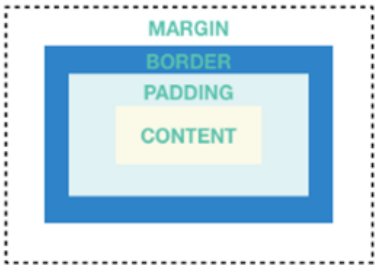


Container

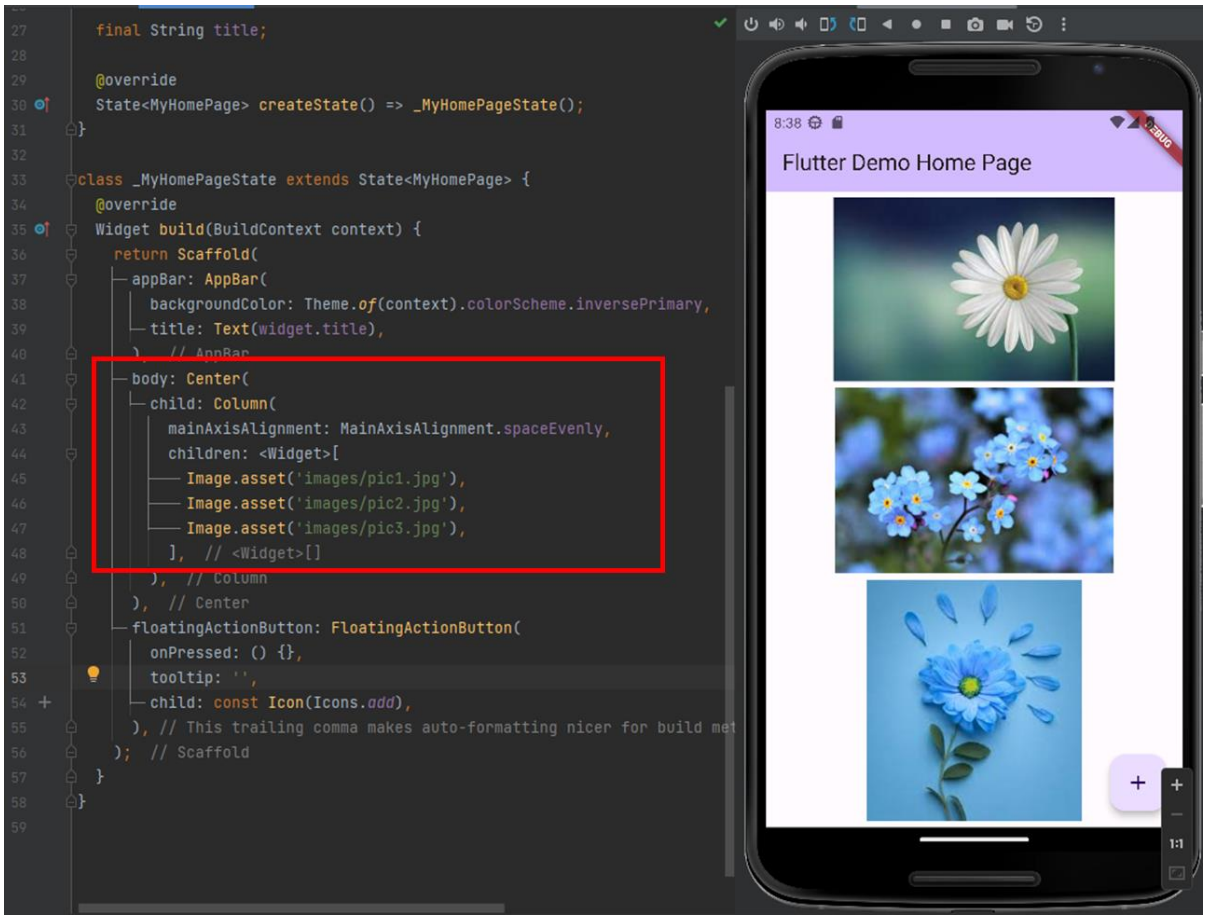
Many layouts make liberal use of `Container`s to separate widgets using padding, or to add borders or margins. You can change the device's background by placing the entire layout into a `Container` and changing its background color or image.

Summary (Container)

- Add padding, margins, borders
- Change background color or image
- Contains a single child widget, but that child can be a Row, Column, or even the root of a widget tree

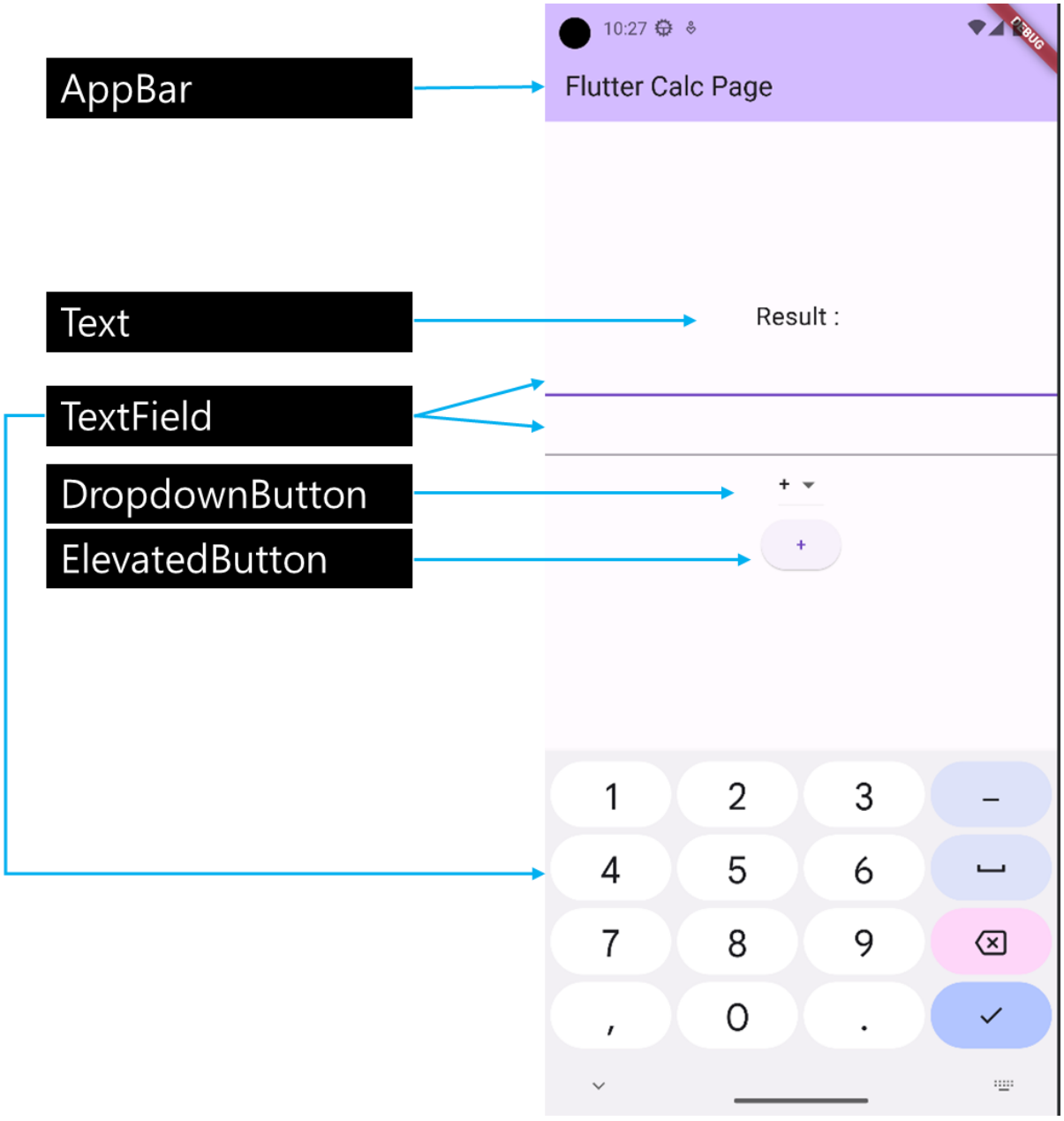


5-8. Image Viewer



[실습]

5-9. Calculator



Dropdownbutton #1

```
// DropdownMenuItem List
final List<DropdownMenuItem<String>> _dropdownmenuitems = const [
    DropdownMenuItem(value: '+', child: Text('+')),
    DropdownMenuItem(value: '-', child: Text('-')),
    DropdownMenuItem(value: 'x', child: Text('x')),
    DropdownMenuItem(value: '/', child: Text('/'))
  ];
// value : 선택되어질때의 값
// child : 목록에 표시되어질 위젯
// DropdownButton Widget
DropdownButton<String>(
                items: _dropdownmenuitems,
                value: buttonText,
                onChanged: (String? value) {
                  setState(() {
                    buttonText = value;
                  });
                },
              ),
```

Dropdownbutton #2

```
// DropdownMenuItem List
final List<String> btxtlist = <String>['+', '-', 'x', '/'];
// DropdownButton Widget
DropdownButton<String>(
            value: buttonText,
            items: btxtlist.map<DropdownMenuItem<String>>((String value) {
              return DropdownMenuItem<String>(
                value: value,
                child: Text(value),
              );
            }).toList(),
            onChanged: (String? value) {
              setState(() {
                buttonText = value;
              });
            },
          ),
```

main.dart

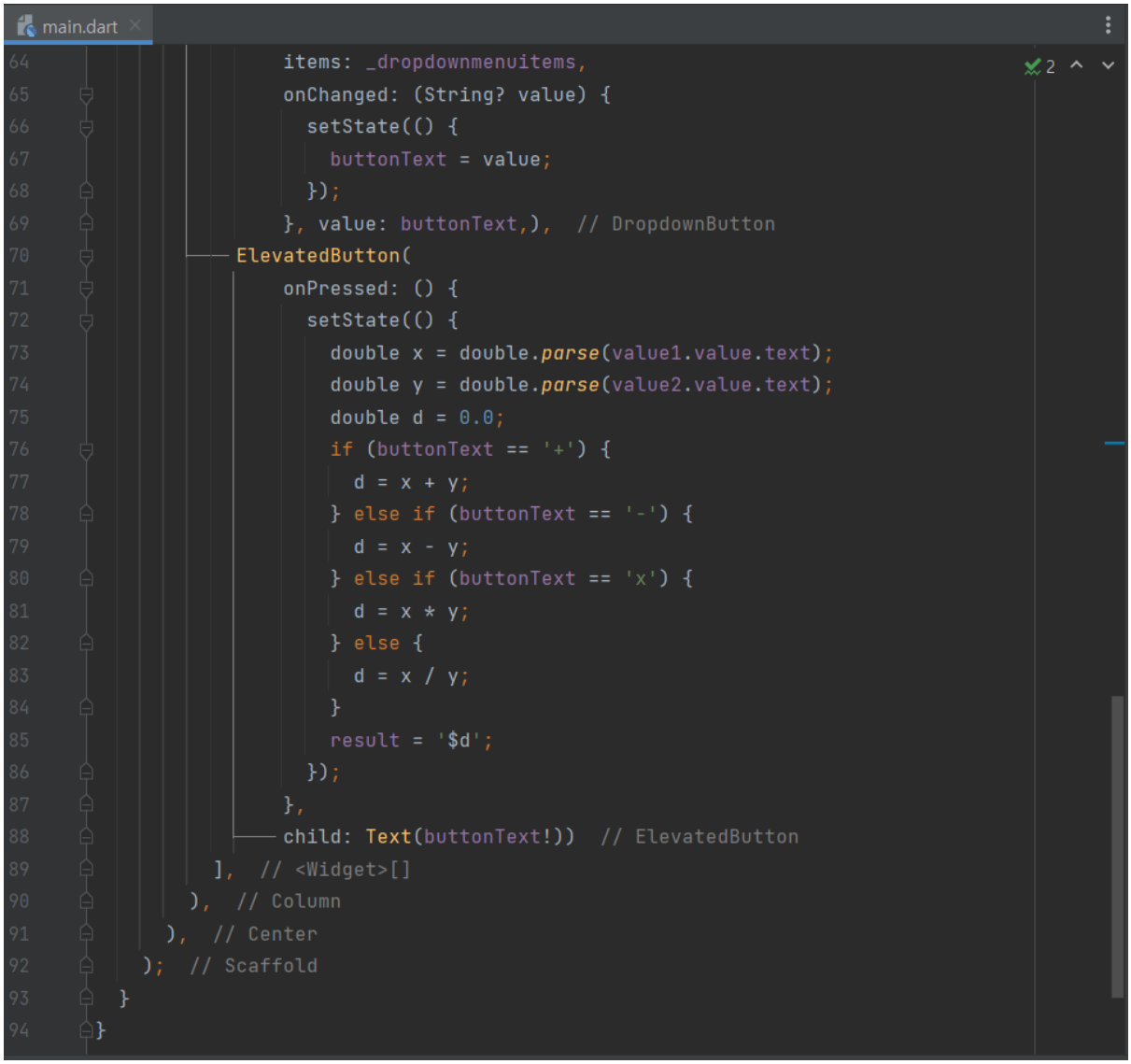
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const Calc());
}
class Calc extends StatelessWidget {
  const Calc({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Calculator',
      theme: ThemeData(),
      home: const MyCalc(),
    );  // MaterialApp
  }
}

class MyCalc extends StatefulWidget {
  const MyCalc({Key? key}) : super(key: key);

  @override
  State<MyCalc> createState() => _MyCalcState();
}

class _MyCalcState extends State<MyCalc> {
  TextEditingController value1 = TextEditingController();
  TextEditingController value2 = TextEditingController();

  var result = '';
  final List buttontextList = ['+', '-', 'x', '/'];
```

main.dart

```dart
class _MyCalcState extends State<MyCalc> {
  TextEditingController value1 = TextEditingController();
  TextEditingController value2 = TextEditingController();

  var result = '';
  final List buttontextList = ['+', '-', 'x', '/'];
  final List<DropdownMenuItem<String>> _dropdownmenuitems =
      List.empty(growable: true);
  String? buttonText;

  @override
  void initState() {
    // TODO: implement initState
    super.initState();

    for(var item in buttontextList) {
      _dropdownmenuitems.add(DropdownMenuItem(value : item, child: Text(item)));
    }
    buttonText = _dropdownmenuitems[0].value;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Calculator"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

main.dart

```dart

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Calculator"),
      ),  // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text("Result : $result"),
            TextField(controller: value1, keyboardType: TextInputType.number),
            TextField(
              controller: value2,
              keyboardType: TextInputType.number,
            ),  // TextField
            DropdownButton(
                items: _dropdownmenuitems,
                onChanged: (String? value) {
                  setState(() {
                    buttonText = value;
                  });
                }, value: buttonText,),  // DropdownButton
            ElevatedButton(
                onPressed: () {
                  setState(() {
                    double x = double.parse(value1.value.text);
                    double y = double.parse(value2.value.text);
                    double d = 0.0;
                    if (buttonText == '+') {
```


main.dart
items: _dropdownmenuitems,
onChanged: (String? value) {
setState(() {
buttonText = value;
});
}, value: buttonText,), // DropdownButton
ElevatedButton(
onPressed: () {
setState(() {
double x = double.parse(value1.value.text);
double y = double.parse(value2.value.text);
double d = 0.0;
if (buttonText == '+') {
d = x + y;
} else if (buttonText == '-') {
d = x - y;
} else if (buttonText == 'x') {
d = x * y;
} else {
d = x / y;
}
result = '$d';
});
},
child: Text(buttonText!)) // ElevatedButton
], // <Widget>[]
), // Column
), // Center
); // Scaffold
}
}

[실습]

https://api.flutter.dev/flutter/material/TabBar-class.html

5-10. Tabbar

TIP)

단축키 : stful + tab or stless + tab

Hello.dart → Stateless widget

Calc.dart → Stateful widget

코드정리: Ctrl+alt+L

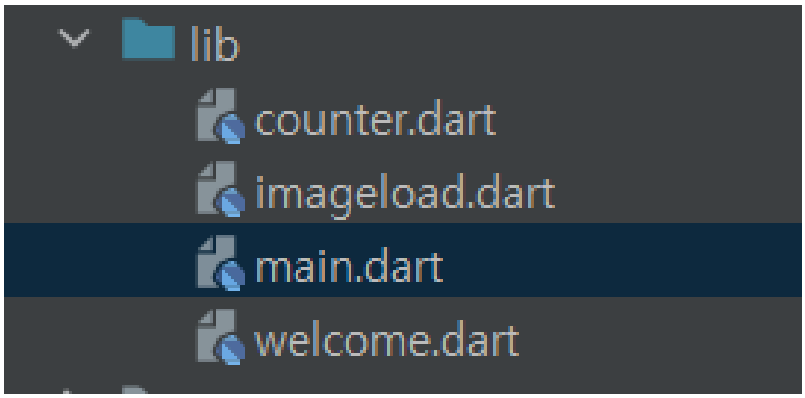
각각 실행되는 4 개의 dart 파일을 만드세요

모두 stateful 로 구성 바랍니다.

1. counter.dart : Default Program – counter 예제
2. imageload.dart : Image Viewer - Flutter 로고 보이게
3. welcome.dart : ElevatedButton 두개 만들고 하나 클릭하면 "Welcome" Text 나오고 다른 하나 누르면 Text Clear.
4. main.dart – "Main Page" Text 출력 →여기에 Tab 구성할 것임

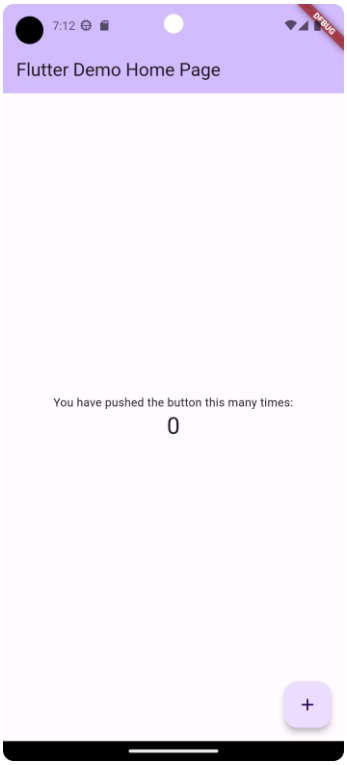
Remind :

Main() → runApp() →stateless MaterialApp() → home : → stateful Scaffold() → body :



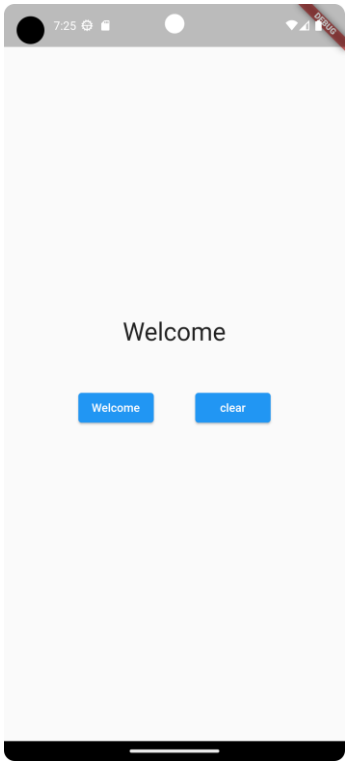
main.dart

counter.dart



imageload.dart

welcome.dart



main.dart

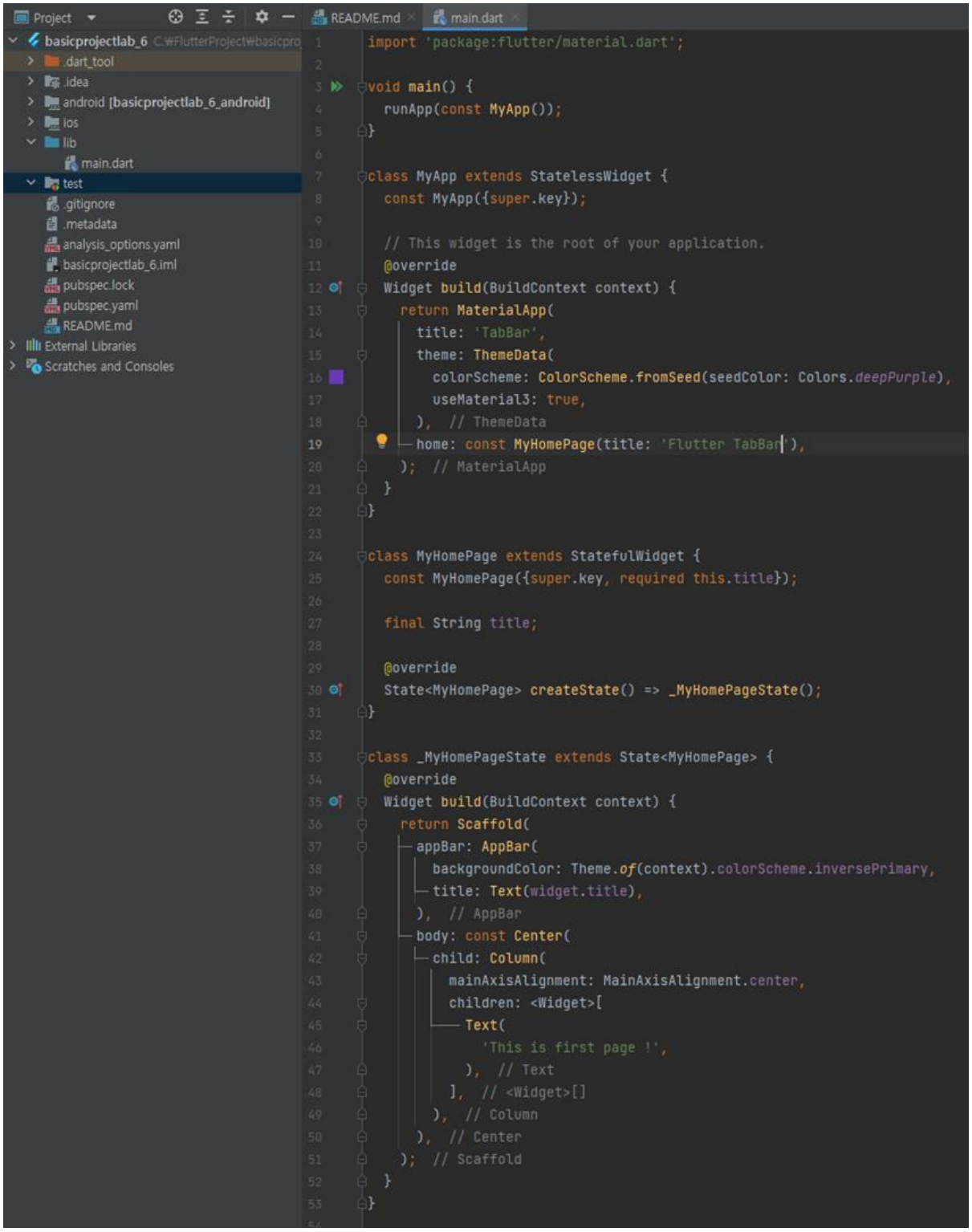
```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const TabEx());
}

class TabEx extends StatelessWidget {
  const TabEx({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: TabExHome(),
    );  // MaterialApp
  }
}

class TabExHome extends StatefulWidget {
  const TabExHome({super.key});

  @override
  State<TabExHome> createState() => _TabExHomeState();
}

class _TabExHomeState extends State<TabExHome> {
  @override
  Widget build(BuildContext context) {
    return const Scaffold(
      body: Center(
        child: Text("TabExHome"),
      ),  // Center
    );  // Scaffold
  }
}

```

imageload.dart

```
child: Image.network('https://sceel.io/wp-content/uploads/2019/10/blog-flutter-header.jpeg'),
```

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(const ImageLoad());
}

class ImageLoad extends StatelessWidget {
  const ImageLoad({super.key});

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: ImageLoadHome(),
    );  // MaterialApp
  }
}

class ImageLoadHome extends StatefulWidget {
  const ImageLoadHome({super.key});

  @override
  State<ImageLoadHome> createState() => _ImageLoadHomeState();
}

class _ImageLoadHomeState extends State<ImageLoadHome> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Image.network('https://sceel.io/wp-content/uploads/20
      ),  // Center
    );  // Scaffold
  }
}

```

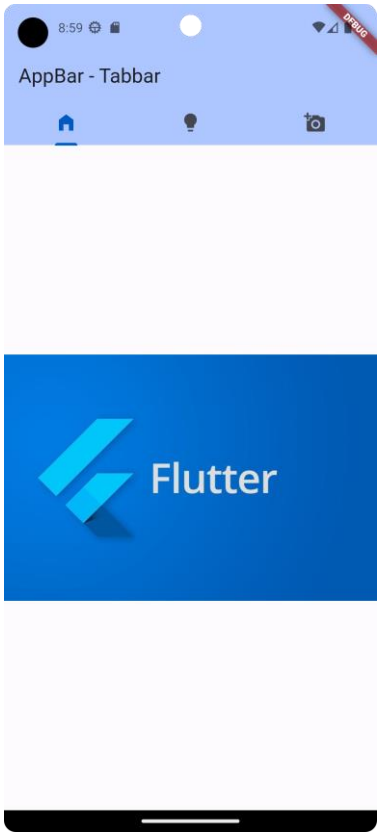
welcome.dart



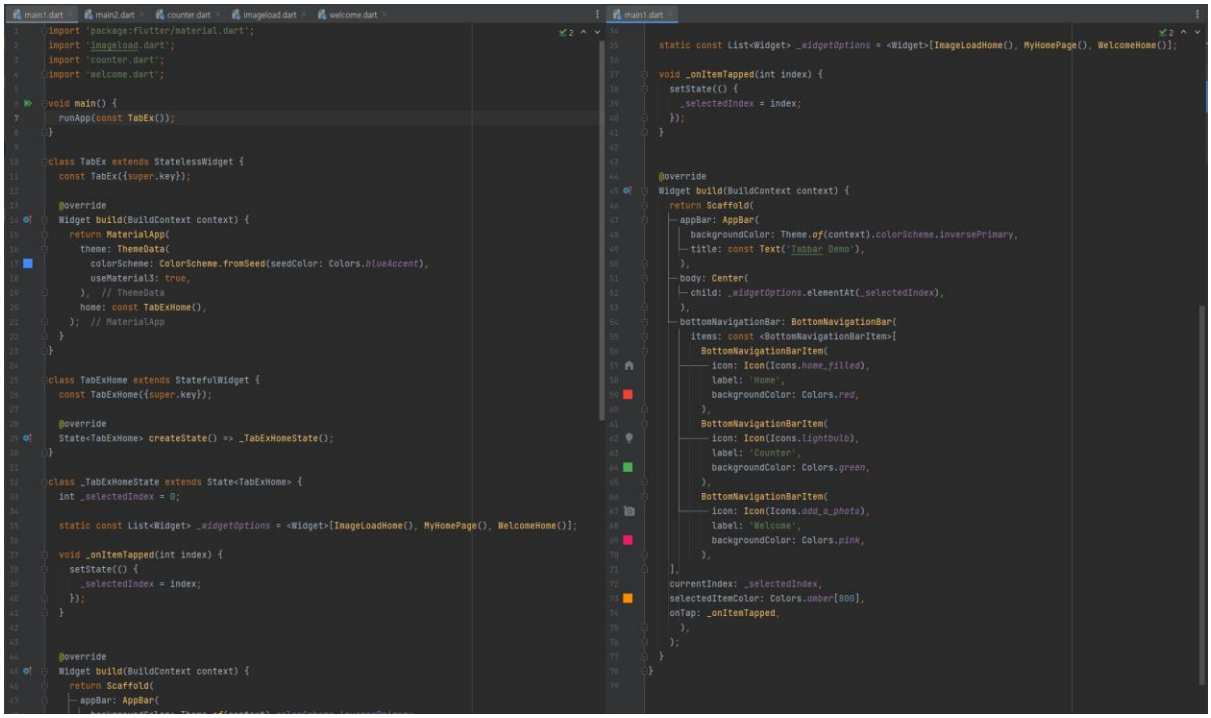
TabBar – DefaultTabController

```dart
import 'package:flutter/material.dart';
import 'imageload.dart';
import 'counter.dart';
import 'welcome.dart';

void main() {
  runApp(const TabEx());
}

class TabEx extends StatelessWidget {
  const TabEx({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blueAccent),
        useMaterial3: true,
      ),  // ThemeData
      home: const TabExHome(),
    );  // MaterialApp
  }
}

class TabExHome extends StatefulWidget {
  const TabExHome({super.key});

  @override
  State<TabExHome> createState() => _TabExHomeState();
}

class _TabExHomeState extends State<TabExHome> {
  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 3,
      child: Scaffold(
        appBar: AppBar(
          backgroundColor: Theme.of(context).colorScheme.inversePrimary,
          title: const Text('AppBar - Tabbar'),
          bottom: const TabBar(
            tabs: <Widget>[
              Tab(icon: Icon(Icons.home_filled)),
              Tab(icon: Icon(Icons.lightbulb)),
              Tab(icon: Icon(Icons.add_a_photo)),
            ],  // <Widget>[]
          ),  // TabBar
```

```dart
class TabEx extends StatelessWidget {
  const TabEx({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blueAccent),
        useMaterial3: true,
      ),
      home: const TabExHome(),
    );
  }
}

class TabExHome extends StatefulWidget {
  const TabExHome({super.key});

  @override
  State<TabExHome> createState() => _TabExHomeState();
}

class _TabExHomeState extends State<TabExHome> {
  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 3,
      child: Scaffold(
        appBar: AppBar(
          backgroundColor: Theme.of(context).colorScheme.inversePrimary,
          title: const Text('AppBar - Tabbar'),
          bottom: const TabBar(
            tabs: <Widget>[
              Tab(icon: Icon(Icons.home_filled)),
              Tab(icon: Icon(Icons.lightbulb)),
              Tab(icon: Icon(Icons.add_a_photo)),
            ],
          ),
        ),
        body: const TabBarView(
          children: <Widget>[ImageLoadHome(), MyHomePage(), WelcomeHome()],
        ),
      ),
    );
  }
}
```



TabBar – bottomNavigationBar



```dart
import 'package:flutter/material.dart';
import 'imageload.dart';
import 'counter.dart';
import 'welcome.dart';

void main() {
  runApp(const TabEx());
}

class TabEx extends StatelessWidget {
  const TabEx({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blueAccent),
        useMaterial3: true,
      ),  // ThemeData
      home: const TabExHome(),
    );  // MaterialApp
  }
}

class TabExHome extends StatefulWidget {
  const TabExHome({super.key});

  @override
  State<TabExHome> createState() => _TabExHomeState();
}

class _TabExHomeState extends State<TabExHome> {
  int _selectedIndex = 0;

  static const List<Widget> _widgetOptions = <Widget>[ImageLoadHome(), MyHomePage(), WelcomeHome()];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
```

```dart
  static const List<Widget> _widgetOptions = <Widget>[ImageLoadHome(), MyHomePage(), WelcomeHome()];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }


  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: const Text('Tabbar Demo'),
      ),
      body: Center(
        child: _widgetOptions.elementAt(_selectedIndex),
      ),
      bottomNavigationBar: BottomNavigationBar(
        items: const <BottomNavigationBarItem>[
          BottomNavigationBarItem(
            icon: Icon(Icons.home_filled),
            label: 'Home',
            backgroundColor: Colors.red,
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.lightbulb),
            label: 'Counter',
            backgroundColor: Colors.green,
          ),
          BottomNavigationBarItem(
            icon: Icon(Icons.add_a_photo),
            label: 'Welcome',
            backgroundColor: Colors.pink,
          ),
        ],
        currentIndex: _selectedIndex,
        selectedItemColor: Colors.amber[800],
        onTap: _onItemTapped,
      ),
    );
  }
}
```

TabBar - TabController

```dart
import 'package:flutter/material.dart';
import 'imageload.dart';
import 'counter.dart';
import 'welcome.dart';

void main() {
  runApp(const TabEx());
}

class TabEx extends StatelessWidget {
  const TabEx({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.blueAccent),
        useMaterial3: true,
      ),  // ThemeData
      home: const TabExHome(),
    );  // MaterialApp
  }
}

class TabExHome extends StatefulWidget {
  const TabExHome({super.key});

  @override
  State<TabExHome> createState() => _TabExHomeState();
}

class _TabExHomeState extends State<TabExHome>
    with SingleTickerProviderStateMixin {

  TabController? controller;

  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    controller = TabController(length: 3, vsync: this);
  }

  @override
  void dispose() {
    // TODO: implement dispose
    controller!.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: const Text('Tabbar Demo'),
      ),
      body: TabBarView(
        controller: controller,
        children: const <Widget>[ImageLoadHome(), MyHomePage(title: 'Counter',), WelcomeHome()],
      ),
      bottomNavigationBar: TabBar(tabs: const <Tab>[
        Tab(icon: Icon(Icons.home_filled),),
        Tab(icon: Icon(Icons.lightbulb),),
        Tab(icon: Icon(Icons.add_a_photo),),
      ], controller: controller,),
    );
  }
}
```

5-11. Navigator (page route)

https://flutter-ko.dev/docs/cookbook/navigation/named-routes

변경된 버튼 종류

1.FlatButton → TextButton

2.Outline Button → OutlinedButton

3.RaisedButton → ElevatedButton

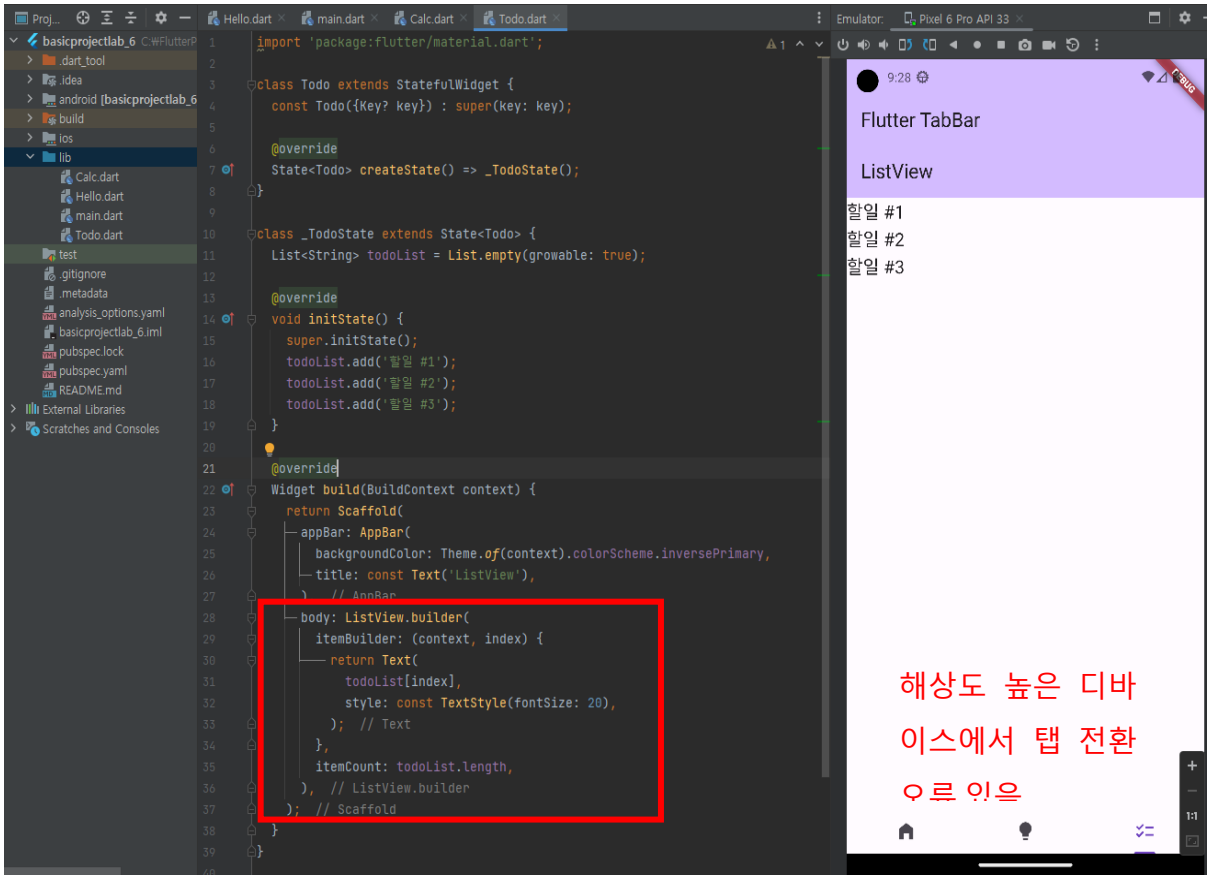
main.dart | FirstScreen.dart | SecondScreen.dart

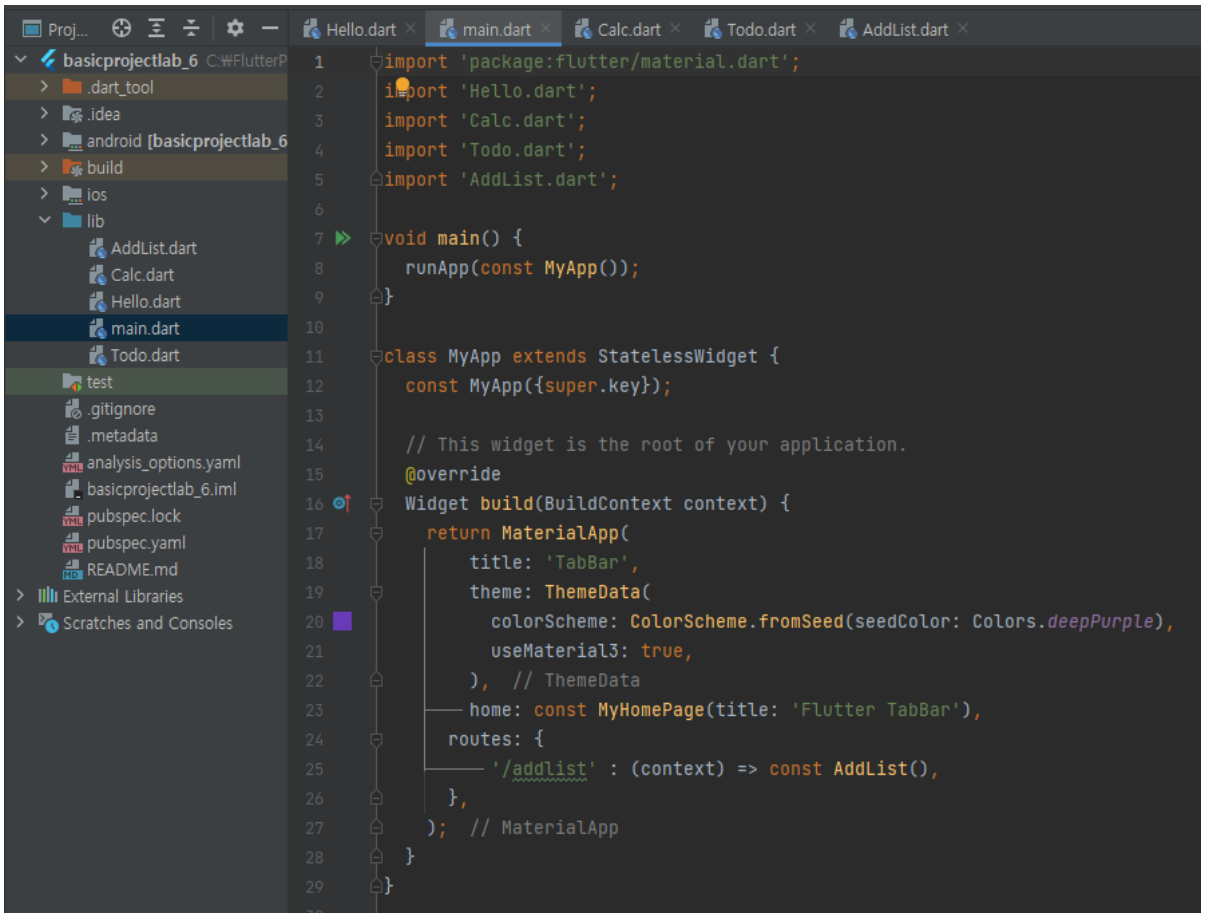
```dart
import 'package:flutter/material.dart';

class FirstScreen extends StatelessWidget {
  const FirstScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Screen'),
      ),  // AppBar
      body: Center(
        child: /*RaisedButton*/ElevatedButton(
          child: const Text('Launch screen'),
          onPressed: () {
            // Named route를 사용하여 두 번째 화면으로 전환합니다.
            Navigator.pushNamed(context, '/second');
          },
        ),  // ElevatedButton
      ),  // Center
    );  // Scaffold
  }
}
```

```dart
import 'package:flutter/material.dart';

class SecondScreen extends StatelessWidget {
  const SecondScreen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Screen'),
      ),  // AppBar
      body: Center(
        child: /*RaisedButton*/ElevatedButton(
          child: const Text('Go back!!'),
          onPressed: () {
            // 현재 route를 스택에서 제거함으로써 첫 번째 스크린으로 되돌아 갑니다.
            Navigator.pop(context);
          },
        ),  // ElevatedButton
      ),  // Center
    );  // Scaffold
  }
}
```

5-12. Listview

```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ),  // AppBar
      body: TabBarView(
        controller: controller,
        children: const <Widget>[Hello(), Calc(), Todo()],
      ),  // TabBarView
      bottomNavigationBar: TabBar(tabs: const <Tab>[
        Tab(icon: Icon(Icons.home_filled)),
        Tab(icon: Icon(Icons.lightbulb)),
        Tab(icon: Icon(Icons.checklist)),
      ],  // <Tab>[]
      controller: controller)  // TabBar
    );  // Scaffold
  }
}
```


Hello.dart
main.dart
Calc.dart
Todo.dart
Emulator: Pixel 6 Pro API 33
9:28
Flutter TabBar
ListView
할일 #1
할일 #2
할일 #3
해상도 높은 디바
이스에서 탭 전환
으로 인을

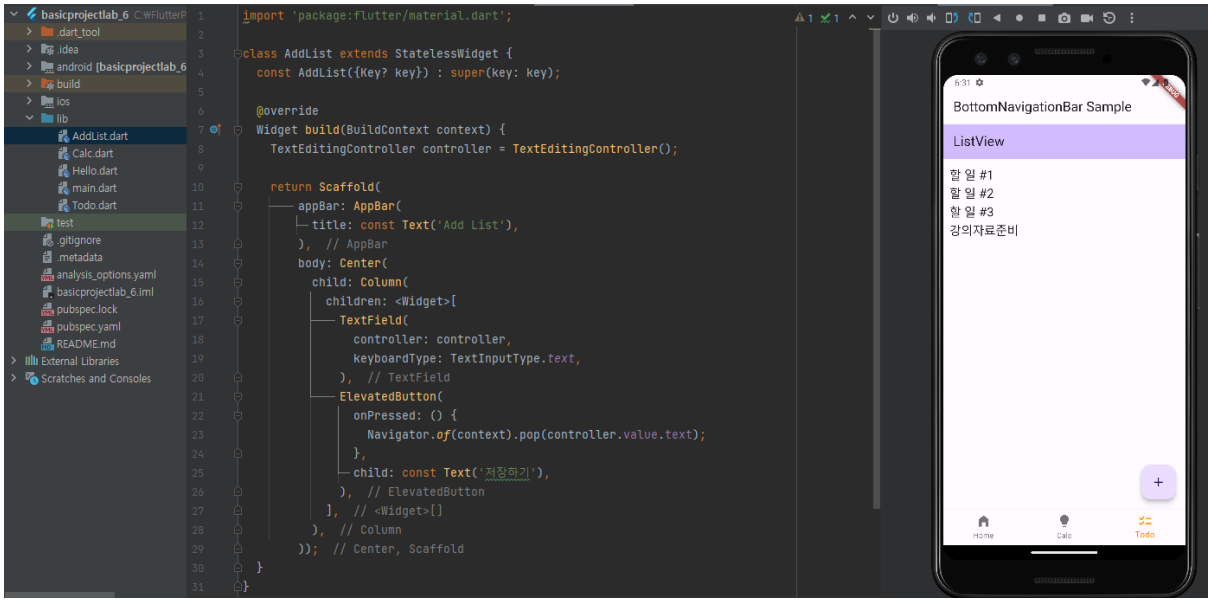

Hello.dart
main.dart
Calc.dart
Todo.dart
AddList.dart
basicprojectlab_6
.dart_tool
.idea
android [basicprojectlab_6
build
ios
lib
AddList.dart
Calc.dart
Hello.dart
main.dart
Todo.dart
test
.gitignore
.metadata
analysis_options.yaml
basicprojectlab_6.iml
pubspec.lock
pubspec.yaml
README.md
External Libraries
Scratches and Consoles

Todo.dart

```dart
import 'package:flutter/material.dart';

class Todo extends StatefulWidget {
  const Todo({Key? key}) : super(key: key);

  @override
  State<Todo> createState() => _TodoState();
}

class _TodoState extends State<Todo> {
  List<String> todoList = List.empty(growable: true);

  @override
  void initState() {
    super.initState();
    todoList.add('할 일 #1');
    todoList.add('할 일 #2');
    todoList.add('할 일 #3');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: const Text('ListView'),
      ),  // AppBar
      body: ListView.builder(
        padding: const EdgeInsets.all(10),
        itemBuilder: (context, index) {
          return Text(
```

Todo.dart

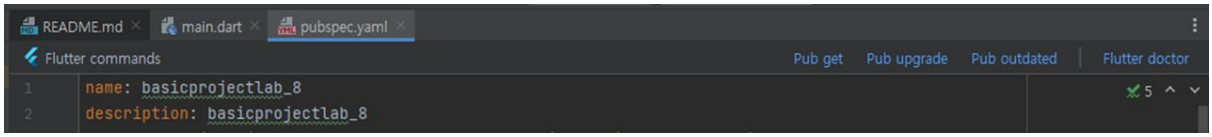
```dart
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: const Text('ListView'),
    ),
    body: ListView.builder(
      padding: const EdgeInsets.all(10),
      itemBuilder: (context, index) {
        return Text(
          todoList[index],
          style: const TextStyle(fontSize: 20),
        );
      },
      itemCount: todoList.length,
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        _addTodo(context);
      },
      child: const Icon(Icons.add),
    ));
}

void _addTodo(BuildContext context) async {
  final result = await Navigator.of(context).pushNamed('/addlist');
  setState(() {
    todoList.add(result as String);
  });
}
}
```

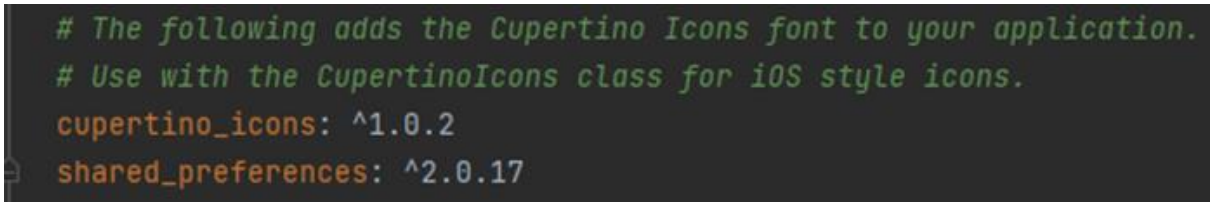


5-13. Shared Preferences

[실습]

Pub.dev 에서 shared_preferences 검색하여 Pubspec.yaml 파일에
패키지 설정하기→Pub get 으로 설치





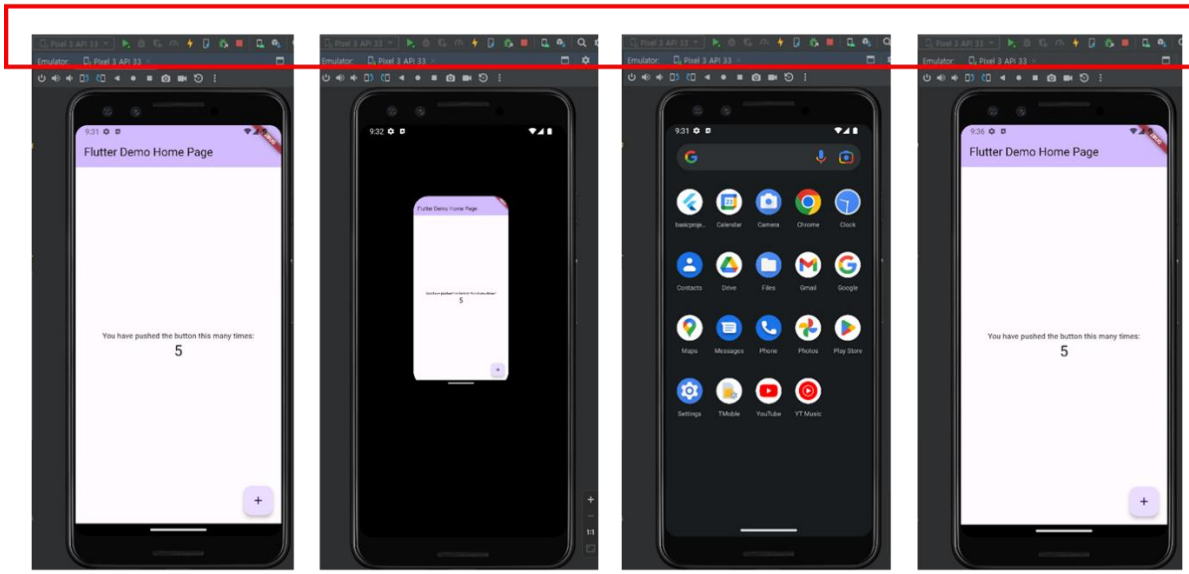
```dart
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

void main() {
  runApp(const MyApp());
}
```

- SharedPreferences 는 간단한 데이터를 저장할 때 사용할 수 있음
- 앱에서만 접근할 수 있는 특수한 목적의 데이터 저장소
- 앱을 실행할때(initState()) 불러오기를 호출하여 데이터를 불러오고,
- incrementCounter() 함수를 호출할 때마다 데이터가 저장되도록 수정하기
- void _setData(int value) async
- void _loadData() async
- 두 함수들은 언제 처리를 완료할지 예측할 수 없으므로 async 를 이용하여 비동기 함수로 만든다.
- SharedPreferences 는 (고유한)키-값 으로 구성
- pref.setInt(key, value); // SharedPreferences method
- var value = pref.getInt(key); // SharedPreferences method
- setState() 함수 안에서의 호출은 State 에서 무언가 변경된 사항이 있음을 Flutter Framework 에 알려주는 역할이다.
- 이로 인해 UI 에 변경된 값이 반영될 수 있도록 build 메소드가 다시 실행된다.

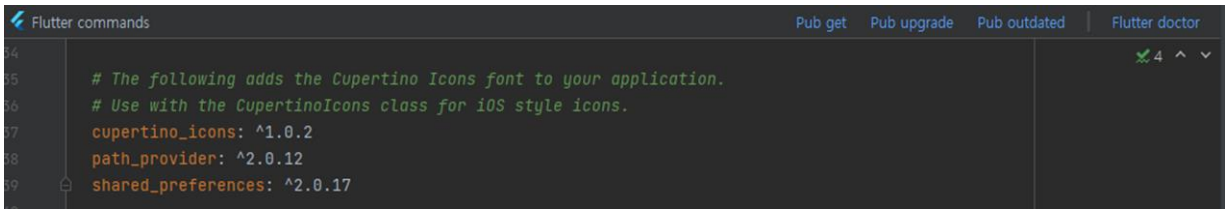
main.dart pubspec.yaml

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _setData(int value) async {
    var key = 'count';
    SharedPreferences pref = await SharedPreferences.getInstance();
    pref.setInt(key, value);
  }

  void _loadData() async {
    var key = 'count';
    SharedPreferences pref = await SharedPreferences.getInstance();
    setState(() {
      var value = pref.getInt(key);
      if (value == null) {
        _counter = 0;
      } else {
        _counter = value;
      }
    });
  }

  @override
  void initState() {
    super.initState();
    _loadData();
  }

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }
```

테스트 하는 도중 에뮬레이터의 상태가 변경되면 안된다.

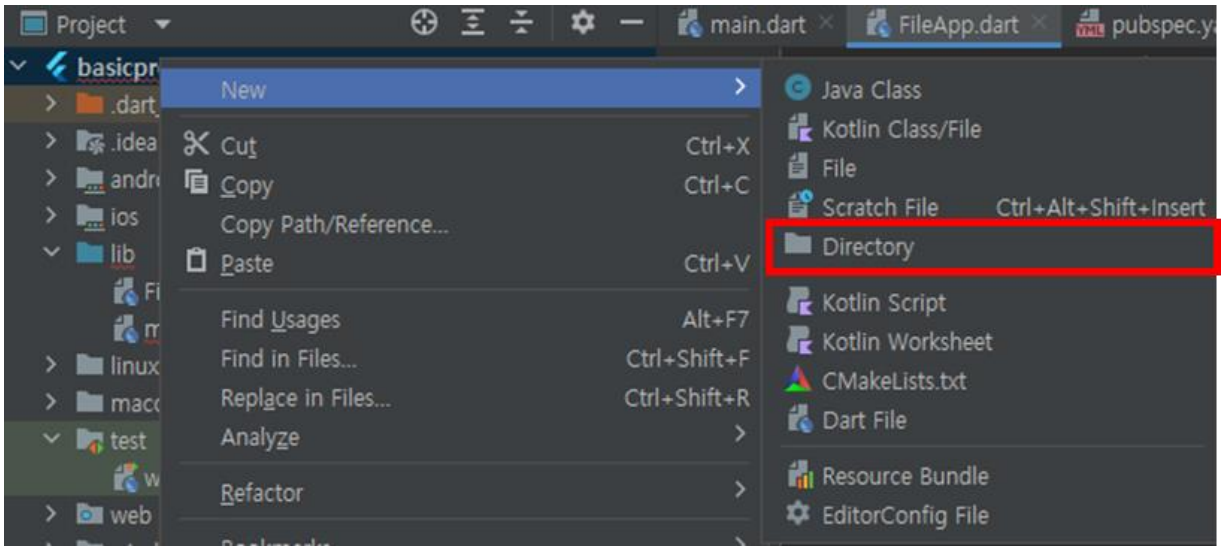


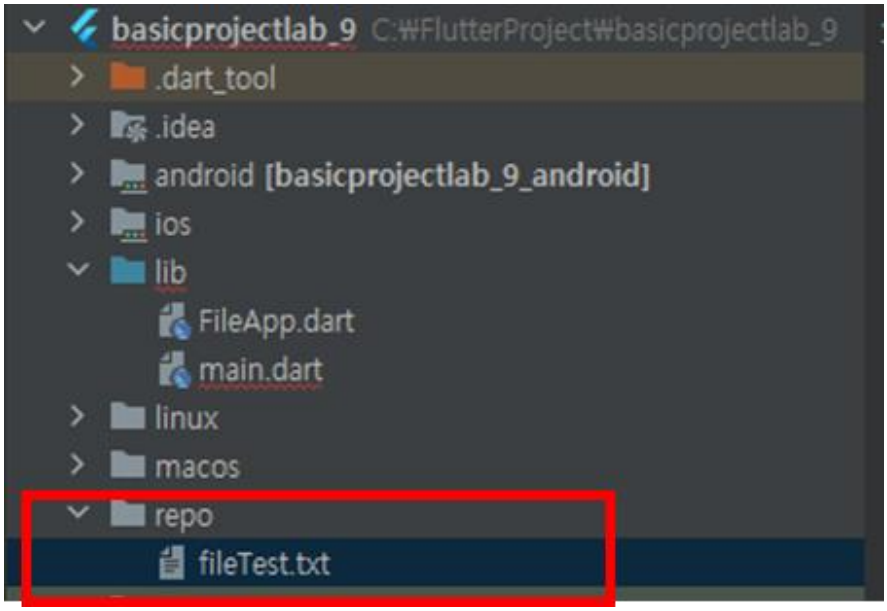
앱을 위쪽으로 스와이프 하여 앱을 종료 후 다시 실행 하여도 count값은 유지 된다.



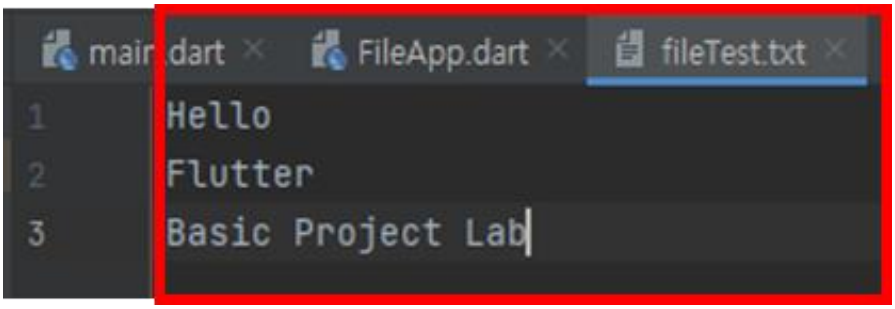
Asset 에 등록된 파일 읽기

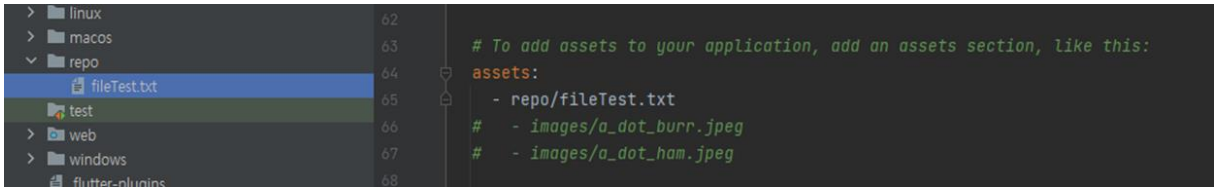
앱 내부 파일에 변경된 내용 저장하기





fileTest.txt 생성





다른 main.dart 에서 FileApp.dart 를 call 하지 않을 경우 FileApp.dart 에 아래 코드를 추가하여야 합니다.

FileApp.dart 에는 Entry Point (main 함수)가 없습니다.

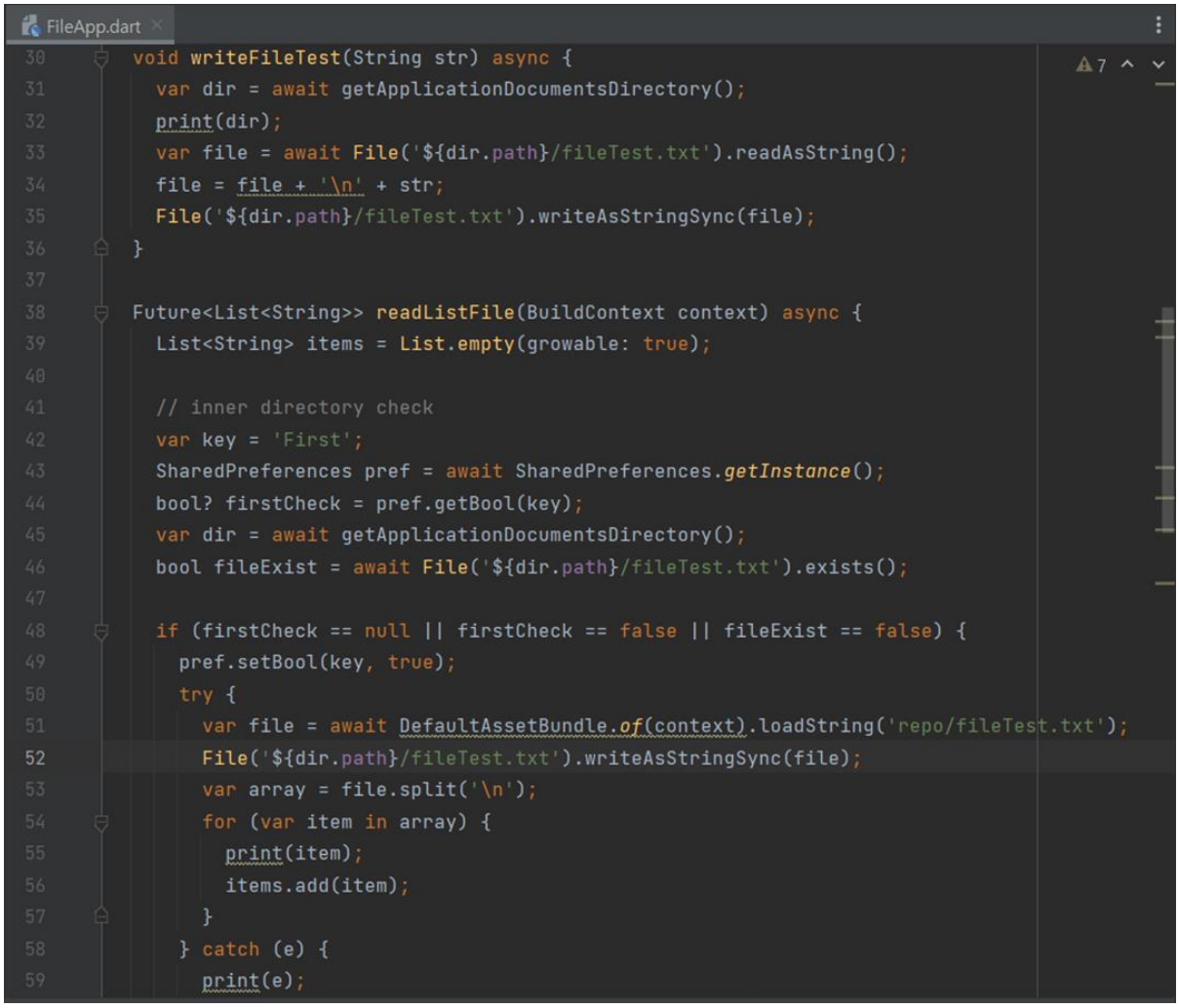
```
void main() {
  runApp(const MaterialApp(
    home: FileApp(), ));
}
```

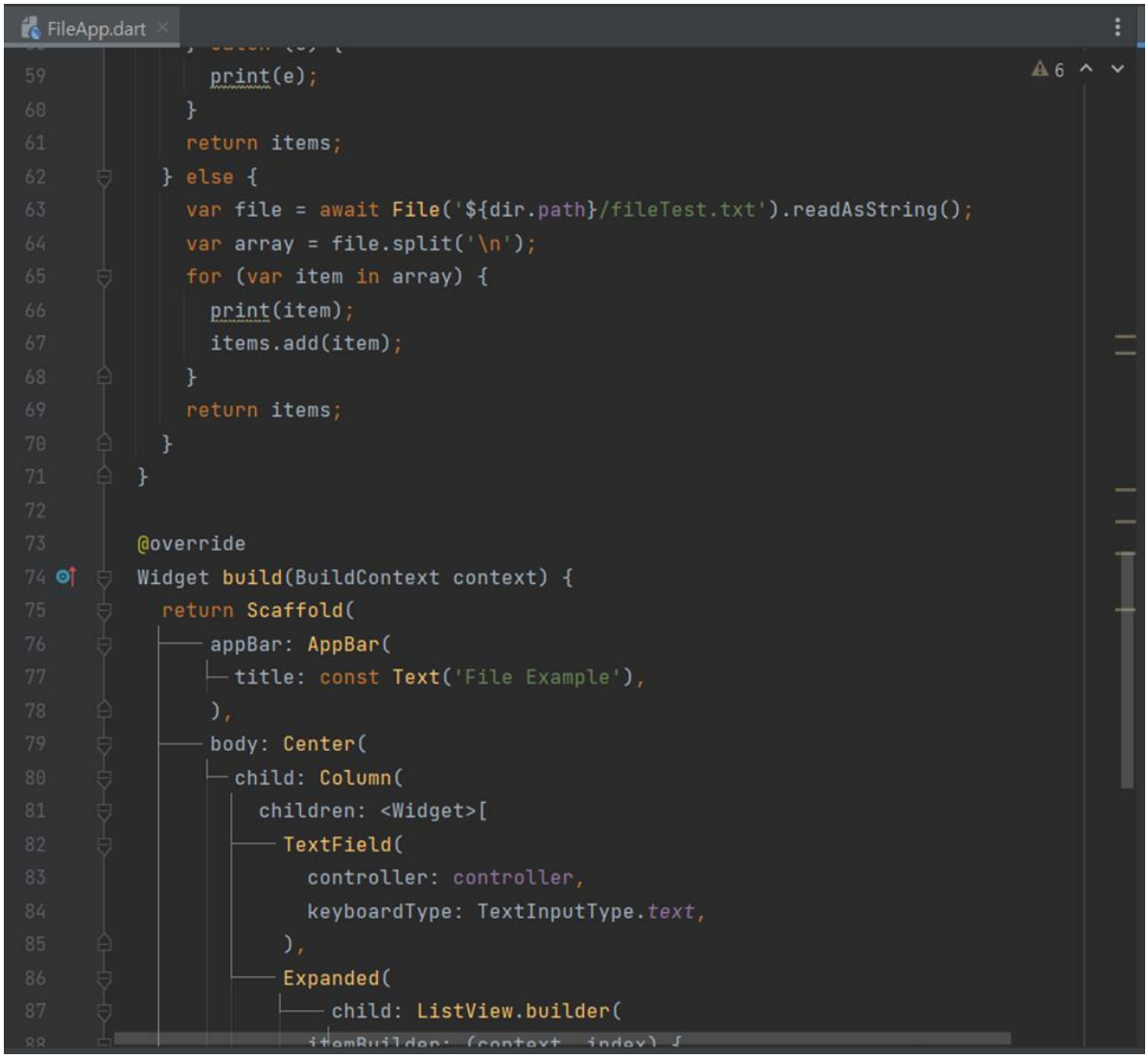
FileApp.dart

```dart
import 'package:flutter/material.dart';
import 'dart:io';
import 'package:path_provider/path_provider.dart';
import 'package:shared_preferences/shared_preferences.dart';

class FileApp extends StatefulWidget {
  const FileApp({super.key});

  @override
  State<StatefulWidget> createState() => _FileApp();
}

class _FileApp extends State<FileApp> {
  List<String> itemList = List.empty(growable: true);
  TextEditingController controller = TextEditingController();

  @override
  void initState() {
    super.initState();
    initData();
  }

  void initData() async {
    var result = await readListFile(context);
    setState(() {
      itemList.addAll(result);
    });
  }

  void writeFileTest(String str) async {
```

이 프로그램은 Asset 에 등록된 File.txt 를 읽은 후 내부 저장소에 새로운 File 을 생성하여 이후 변경되는 내용은 내부 저장소를 이용합니다.

Asset 에 등록되어지는 파일은 Read only 입니다.

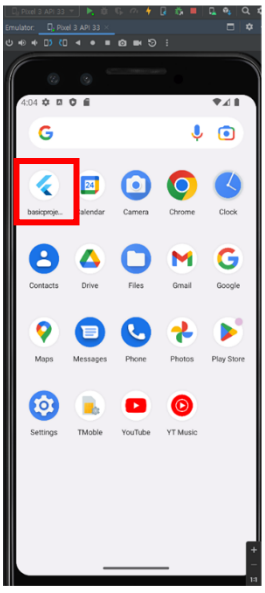



FileApp.dart
print(e);
}
return items;
} else {
var file = await File('${dir.path}/fileTest.txt').readAsString();
var array = file.split('\n');
for (var item in array) {
print(item);
items.add(item);
}
return items;
}
}
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: const Text('File Example'),
),
body: Center(
child: Column(
children: <Widget>[
TextField(
controller: controller,
keyboardType: TextInputType.text,
),
Expanded(
child: ListView.builder(

```dart
FileApp.dart
            keyboardType: TextInputType.text,
          ),
          Expanded(
            child: ListView.builder(
              itemBuilder: (context, index) {
                return Card(
                  child: Center(
                    child: Text(
                      itemList[index],
                      style: const TextStyle(fontSize: 30),
                    ),
                  ),
                );
              },
              itemCount: itemList.length,
            ))
        ],
      ),
    ),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        writeFileTest(controller.value.text);
        setState(() {
          itemList.add(controller.value.text);
        });
      },
      child: const Icon(Icons.add),
    ));
  }
}
```

앱 종료 후 다시 실행하면

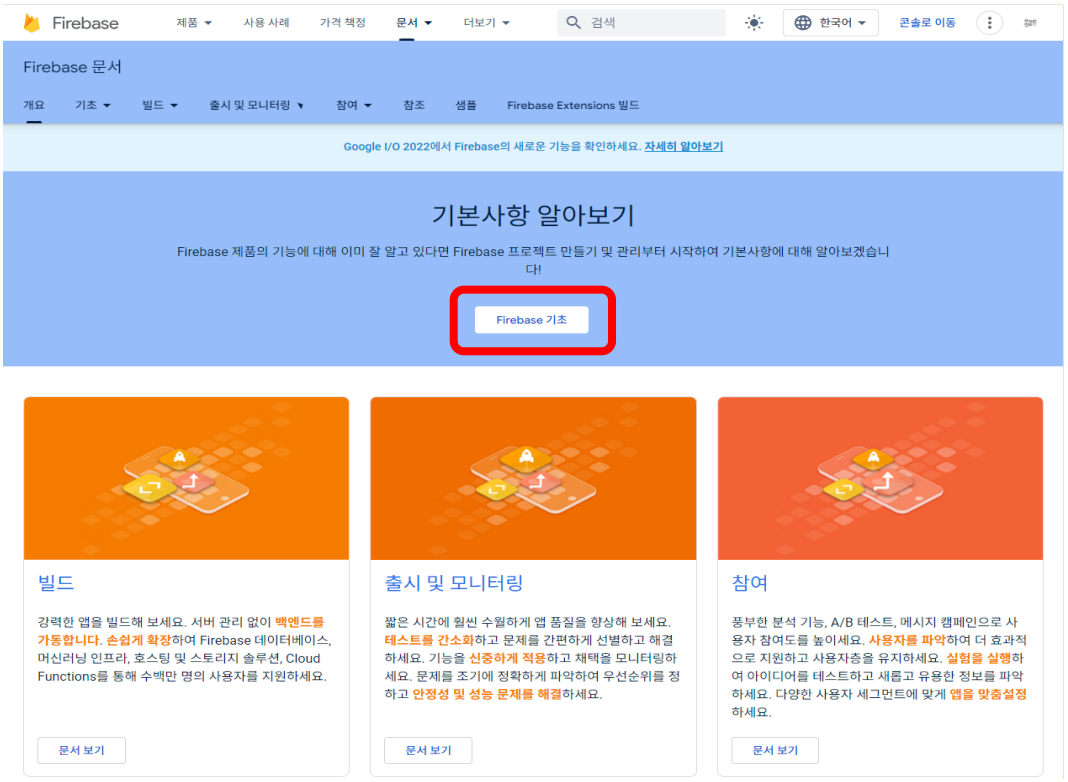
내부 디렉토리에 저장된 리스트를 읽어 옵니다.

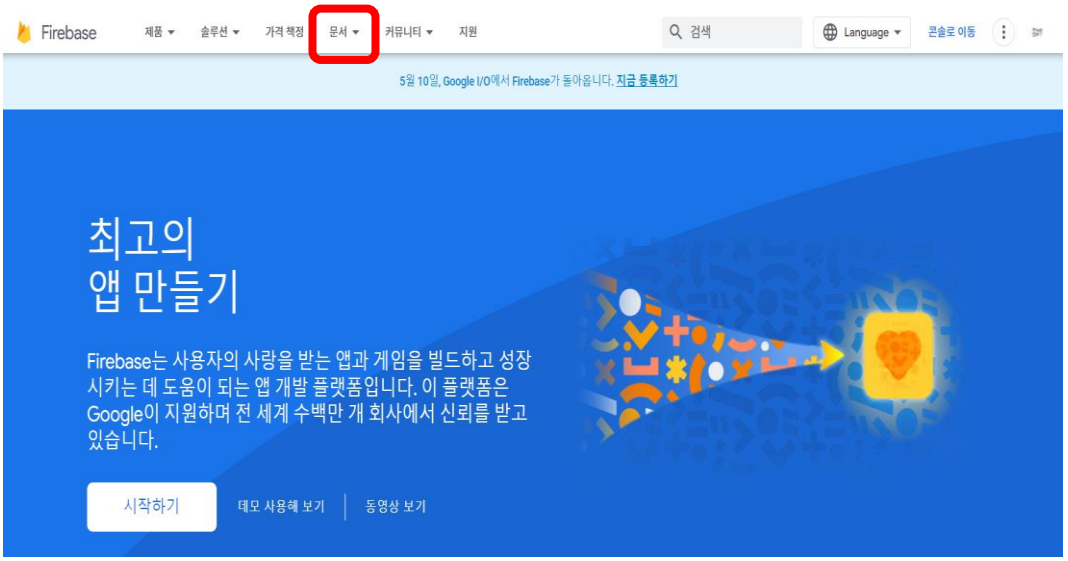


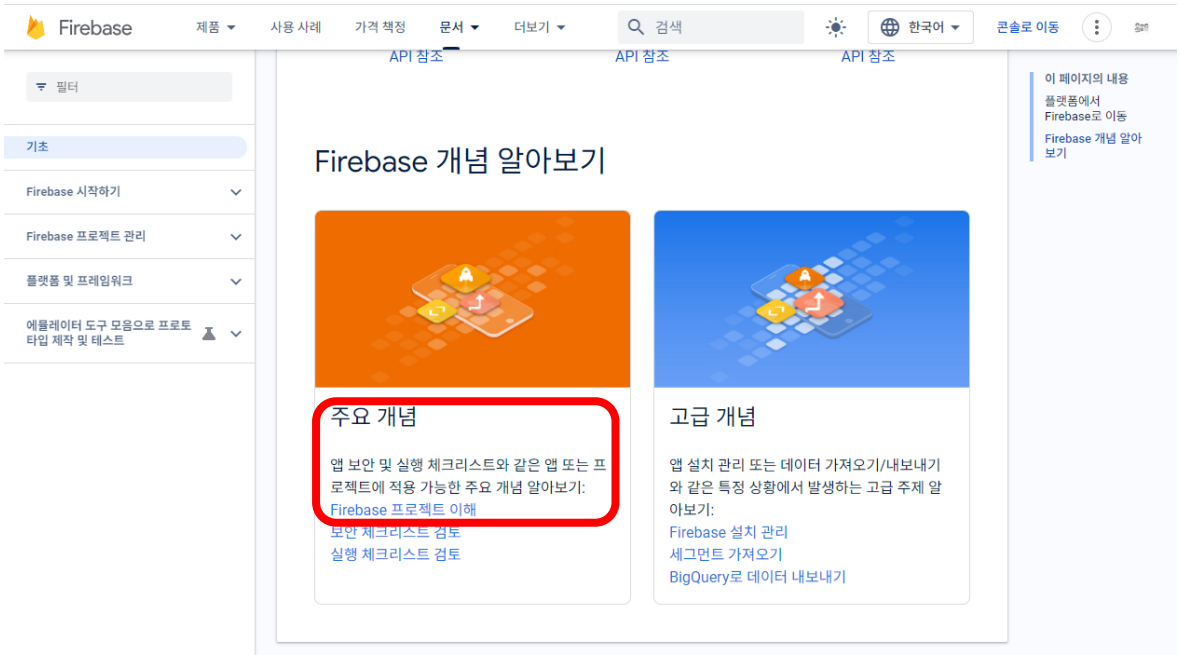
5-14. Firebase

https://firebase.google.com/?hl=ko

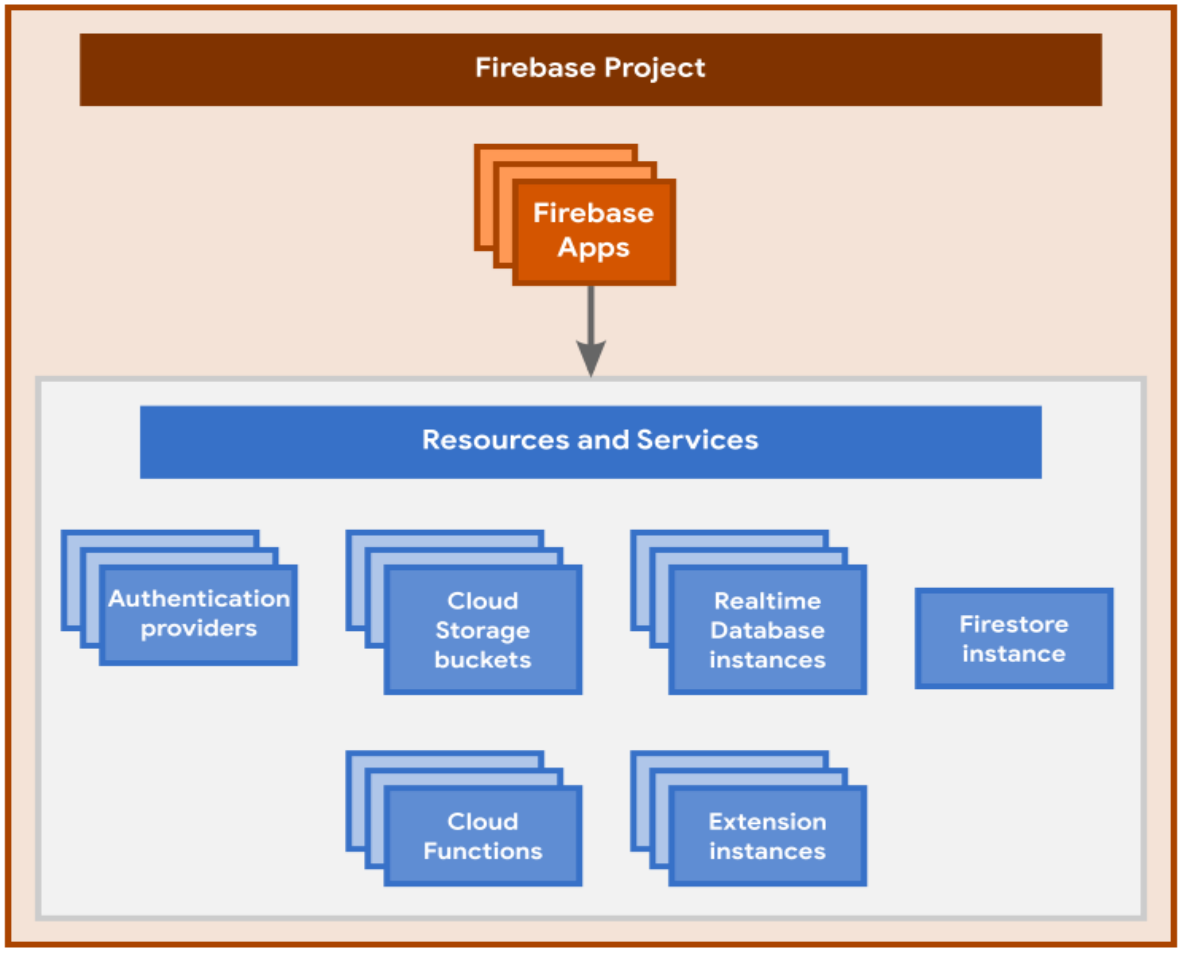
파이어베이스는 모바일 앱이나 웹 애플리케이션을 개발하고 운용할 때 사용할 수 있는 여러가지 서비스를 제공하는 클라우드 플랫폼. 파이어베이스를 이용하면 서버리스(serverless) 앱을 개발하고 운용할 수 있습니다.



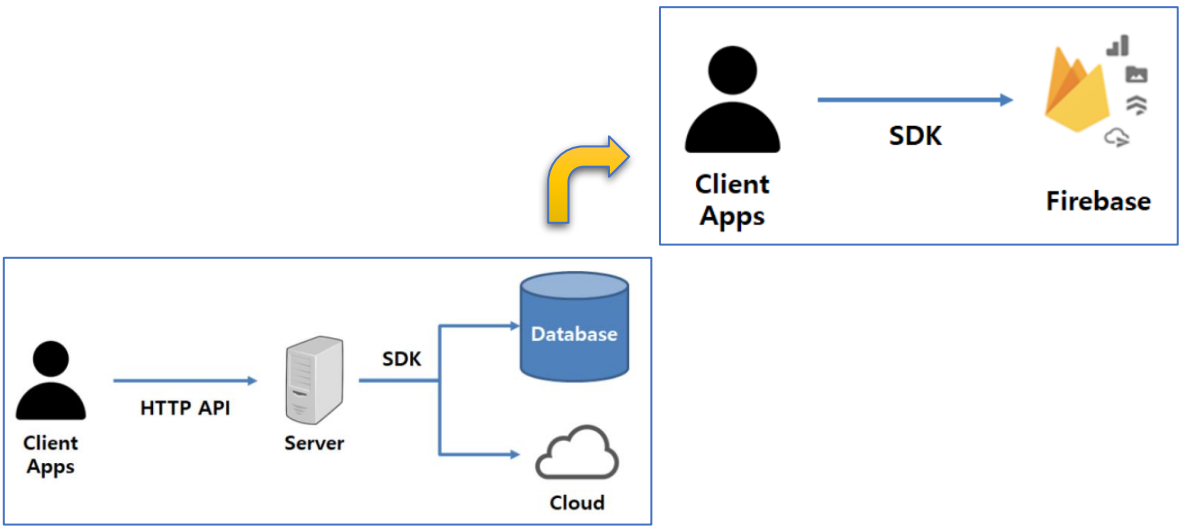




https://firebase.google.com/docs/projects/learn-more?hl=ko

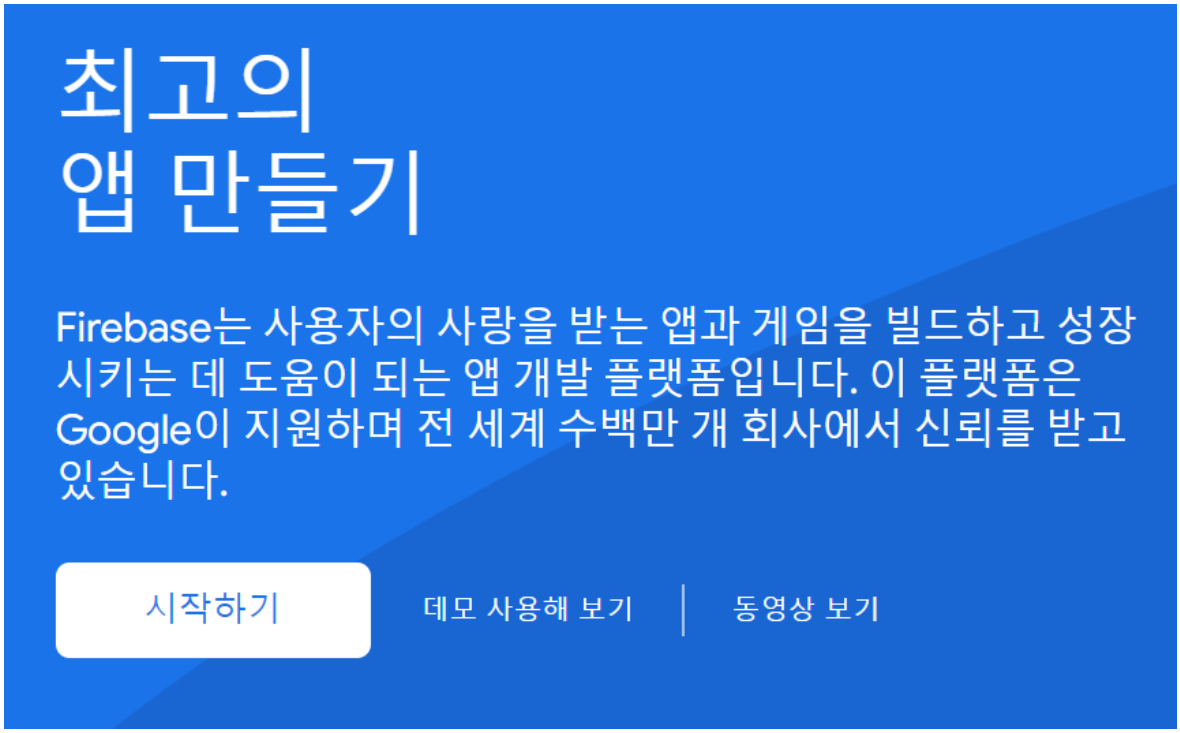


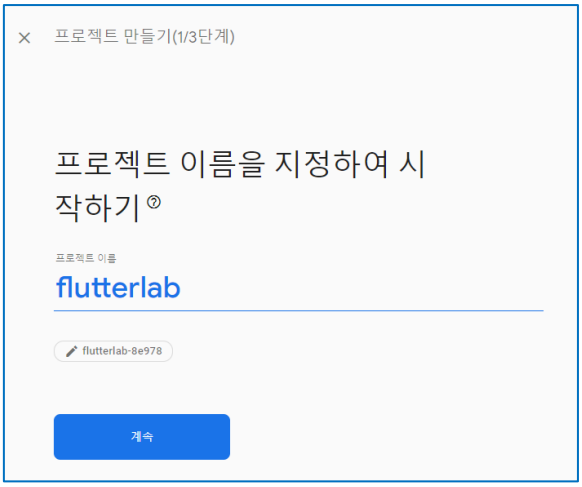
https://blog.naver.com/PostView.naver?blogId=seek316&logNo=222723268021

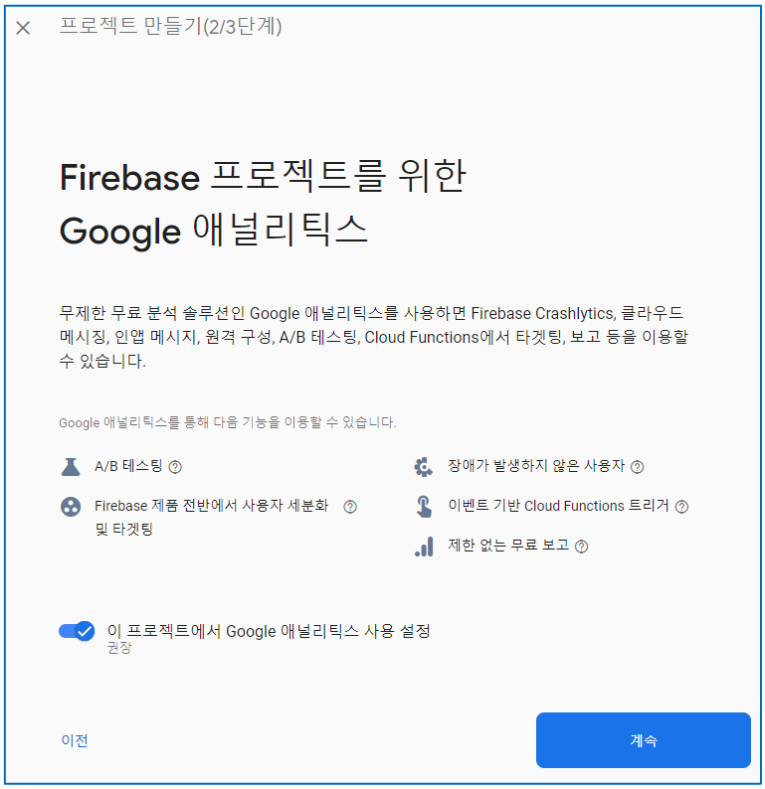


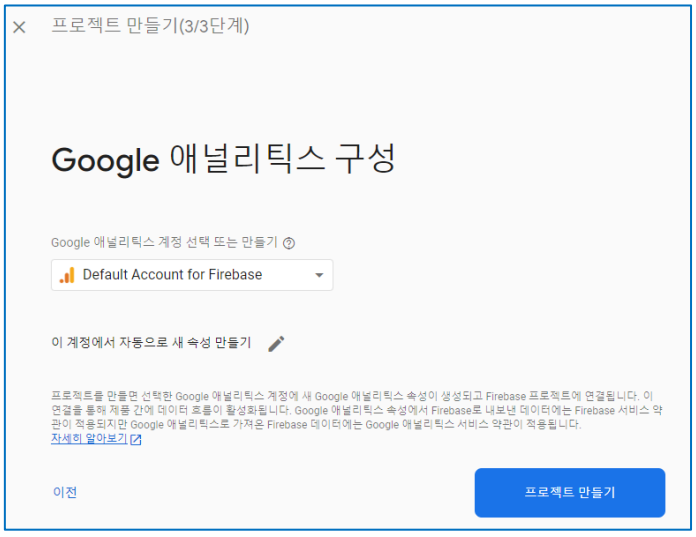
[실습]

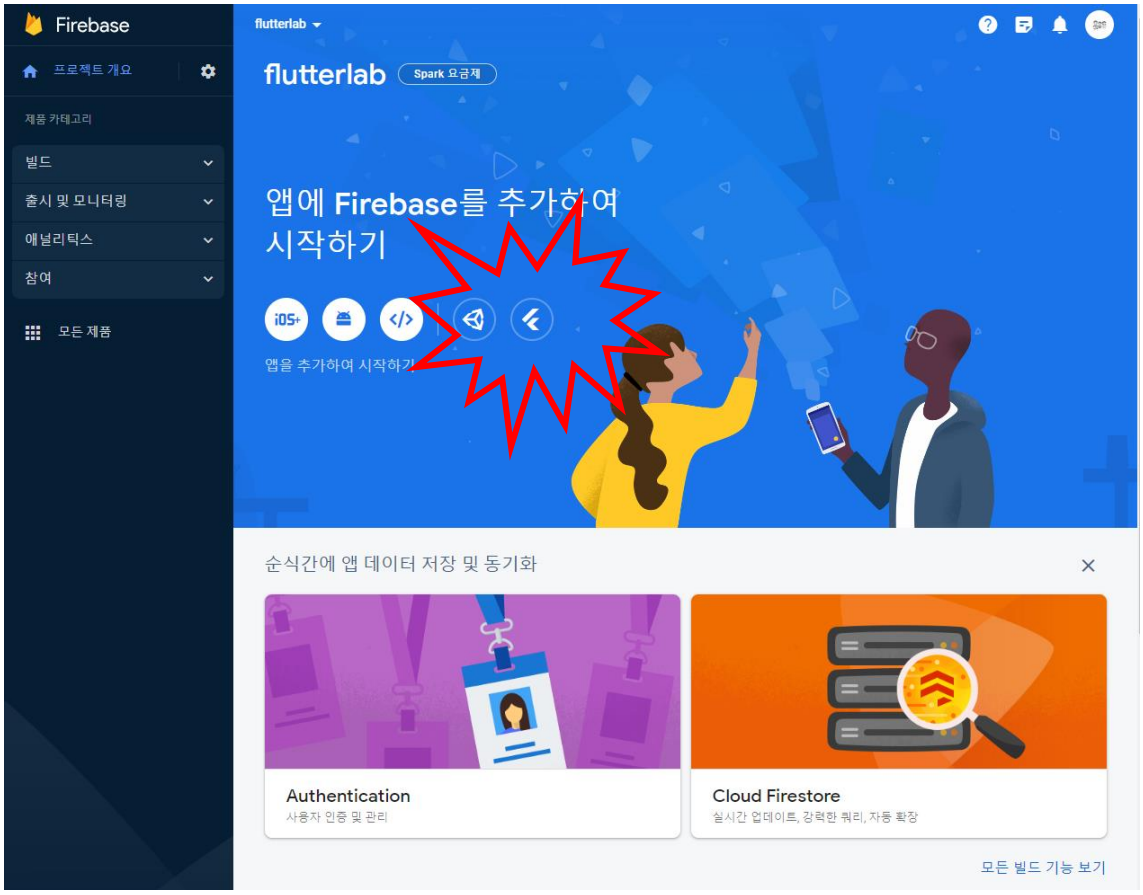
Firebase setting




프로젝트 만들기(1/3단계)
프로젝트 이름을 지정하여 시작하기
프로젝트 이름
flutterlab
flutterlab-8e978
계속

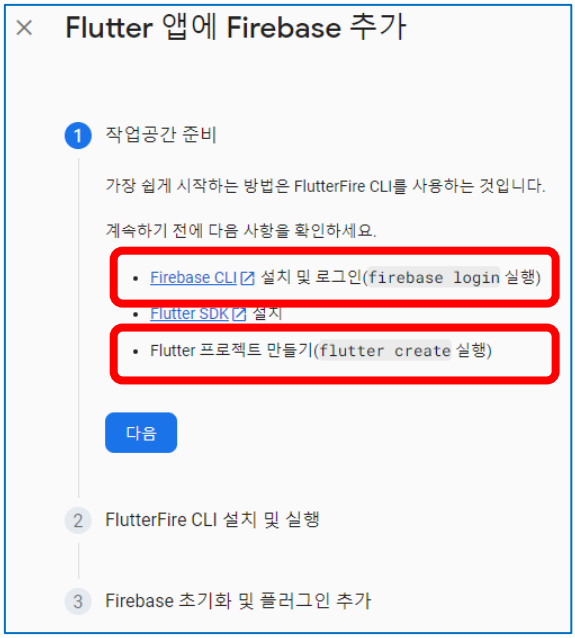

프로젝트 만들기(2/3단계)
Firebase 프로젝트를 위한 Google 애널리틱스
무제한 무료 분석 솔루션인 Google 애널리틱스를 사용하면 Firebase Crashlytics, 클라우드 메시징, 인앱 메시지, 원격 구성, A/B 테스팅, Cloud Functions에서 타겟팅, 보고 등을 이용할 수 있습니다.
Google 애널리틱스를 통해 다음 기능을 이용할 수 있습니다.
A/B 테스팅
장애가 발생하지 않은 사용자
Firebase 제품 전반에서 사용자 세분화 및 타겟팅
이벤트 기반 Cloud Functions 트리거
제한 없는 무료 보고
이 프로젝트에서 Google 애널리틱스 사용 설정
권장
이전
계속


프로젝트 만들기(3/3단계)
Google 애널리틱스 구성
Google 애널리틱스 계정 선택 또는 만들기
Default Account for Firebase
이 계정에서 자동으로 새 속성 만들기
프로젝트를 만들면 선택한 Google 애널리틱스 계정에 새 Google 애널리틱스 속성이 생성되고 Firebase 프로젝트에 연결됩니다. 이 연결을 통해 제품 간에 데이터 흐름이 활성화됩니다. Google 애널리틱스 속성에서 Firebase로 내보낸 데이터에는 Firebase 서비스 약관이 적용되지만 Google 애널리틱스로 가져온 Firebase 데이터에는 Google 애널리틱스 서비스 약관이 적용됩니다.
자세히 알아보기
이전
프로젝트 만들기



Flutter 앱에 Firebase 추가

https://firebase.google.com/docs/flutter/setup?hl=ko&platform=android



Firebase CLI 설치

운영체제, 숙련도 또는 사용 사례에 맞는 방법을 사용하여 Firebase CLI를 설치할 수 있습니다. CLI 설치 방법에 관계없이 동일한 기능 및 `firebase` 명령어에 액세스할 수 있습니다.

Windows | macOS | Linux

Windows

다음 옵션 중 하나를 사용하여 Windows용 Firebase CLI를 설치할 수 있습니다.

옵션	설명	추천 대상
독립 실행형 바이너리	CLI용 독립 실행형 바이너리를 다운로드합니다. 그런 다음 실행 파일에 액세스하여 셸을 열고 `firebase` 명령어를 실행할 수 있습니다.	새 개발자 Node.js를 사용하지 않거나 익숙하지 않은 개발자
npm	npm(노드 패키지 관리자)을 사용하여 CLI를 설치하고 디렉터리에 관계없이 사용 가능한 `firebase` 명령어를 사용 설정합니다.	Node.js를 사용 중인 개발자

독립 실행형 바이너리 | npm

Firebase CLI용 바이너리를 다운로드하고 실행하려면 다음 단계를 따르세요.

1. Windows용 Firebase CLI 바이너리를 다운로드합니다.
2. 바이너리에 액세스하여 셸을 열고 `firebase` 명령어를 실행할 수 있습니다.
3. 계속해서 로그인하여 CLI를 테스트합니다.

로그인하여 Firebase CLI 테스트

CLI를 설치한 후에는 인증해야 합니다. 그러면 Firebase 프로젝트를 나열하여 인증을 확인할 수 있습니다.

1. 다음 명령어를 실행하여 Google 계정으로 Firebase에 로그인합니다.

```
$ firebase login
```

이 명령어는 로컬 머신을 Firebase에 연결하고 Firebase 프로젝트에 대한 액세스 권한을 부여합니다.

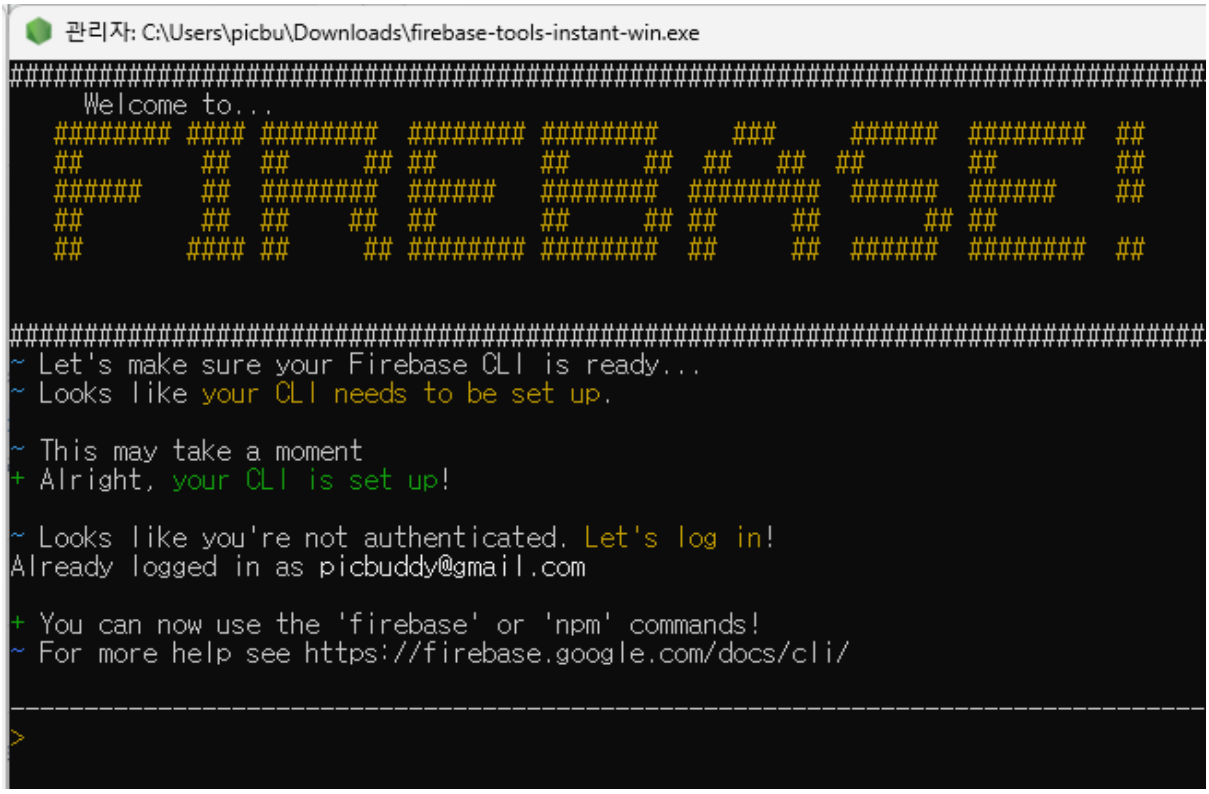
★ **참고:** **`firebase login`** 명령어는 머신에서 **`localhost`**에 연결되는 웹페이지를 엽니다. 원격 머신을 사용 중이며 **`localhost`**에 액세스할 수 없는 경우 **`--no-localhost`** 플래그를 사용하여 명령어를 실행합니다.

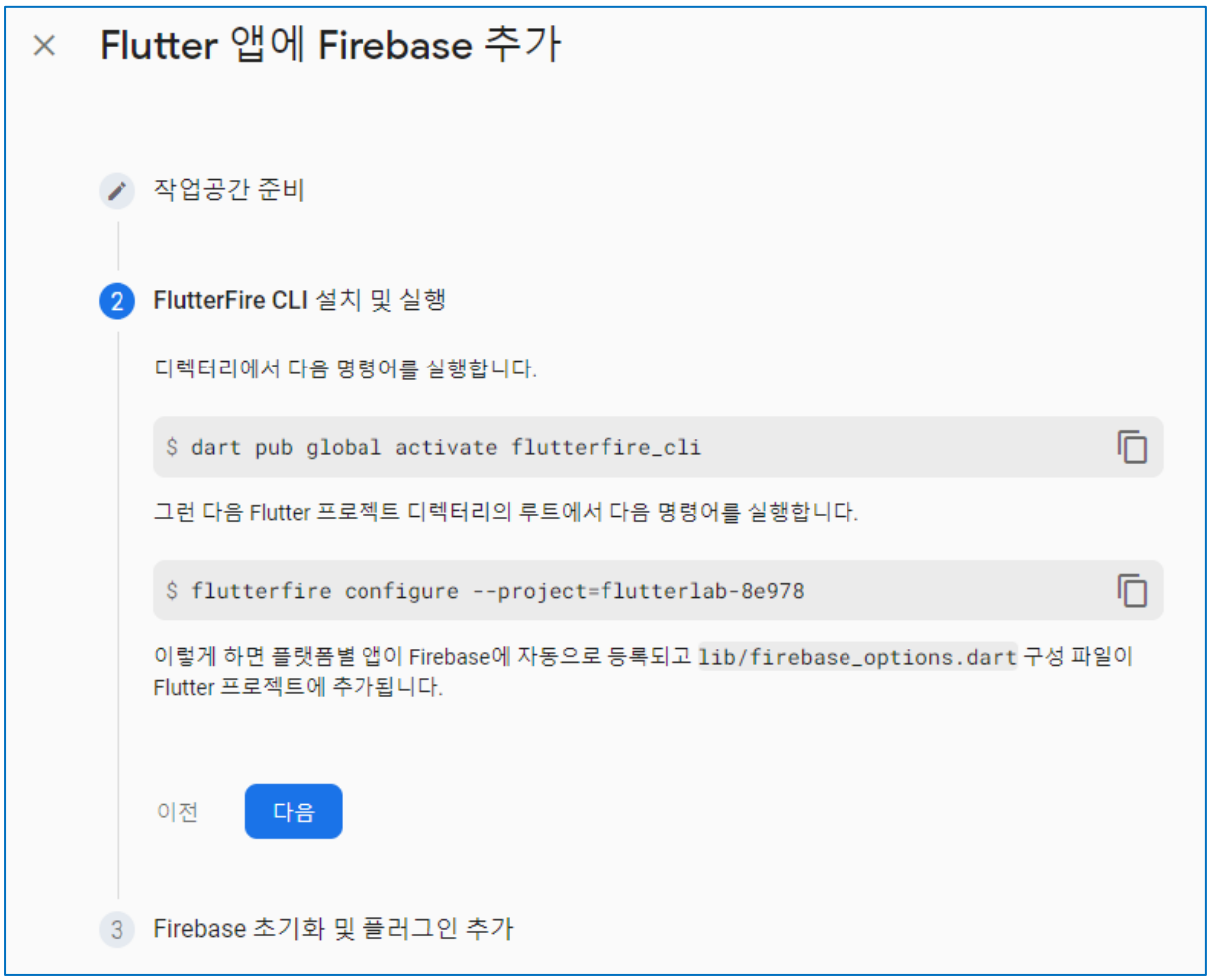
★ **참고:** CI 시스템과 함께 Firebase CLI를 사용할 수도 있습니다.

2. Firebase 프로젝트를 나열하여 CLI가 올바르게 설치되었고 사용자 계정에 액세스하는지 테스트합니다. 다음 명령어를 실행합니다.

```
$ firebase projects:list
```

표시된 목록은 Firebase Console에 나열된 Firebase 프로젝트와 같아야 합니다.


관리자: C:\Users\picbu\Downloads\firebase-tools-instant-win.exe
Welcome to...
~ Let's make sure your Firebase CLI is ready...
~ Looks like your CLI needs to be set up.
~ This may take a moment
+ Alright, your CLI is set up!
~ Looks like you're not authenticated. Let's log in!
Already logged in as picbuddy@gmail.com
+ You can now use the 'firebase' or 'npm' commands!
~ For more help see https://firebase.google.com/docs/cli/
>

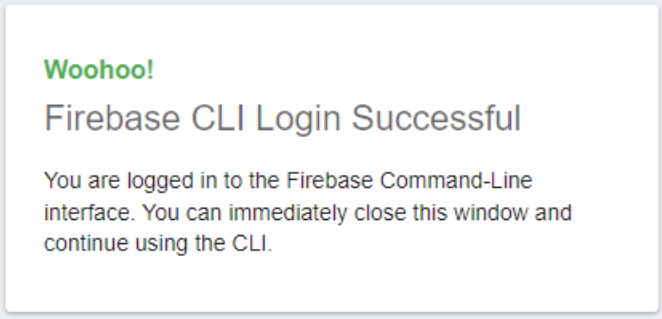

Flutter 앱에 Firebase 추가
작업공간 준비
2
FlutterFire CLI 설치 및 실행
디렉터리에서 다음 명령어를 실행합니다.
$ dart pub global activate flutterfire_cli
그런 다음 Flutter 프로젝트 디렉터리의 루트에서 다음 명령어를 실행합니다.
$ flutterfire configure --project=flutterlab-8e978
이렇게 하면 플랫폼별 앱이 Firebase에 자동으로 등록되고 lib/firebase_options.dart 구성 파일이 Flutter 프로젝트에 추가됩니다.
이전
다음
3
Firebase 초기화 및 플러그인 추가

오류해결

```
> dart pub global activate flutterfire_cli
Package flutterfire_cli is currently active at version 0.2.7.
The package flutterfire_cli is already activated at newest available version.
To recompile executables, first run `dart pub global deactivate flutterfire_cli`.
Installed executable flutterfire.
Warning: Pub installs executables into C:\Users\picbu\AppData\Local\Pub\Cache\bin, which is not on your path.
You can fix that by adding that directory to your system's "Path" environment variable.
A web search for "configure windows path" will show you how.
Activated flutterfire_cli 0.2.7.

> flutterfire configure --project=flutterlab-8e978
'flutterfire'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
```

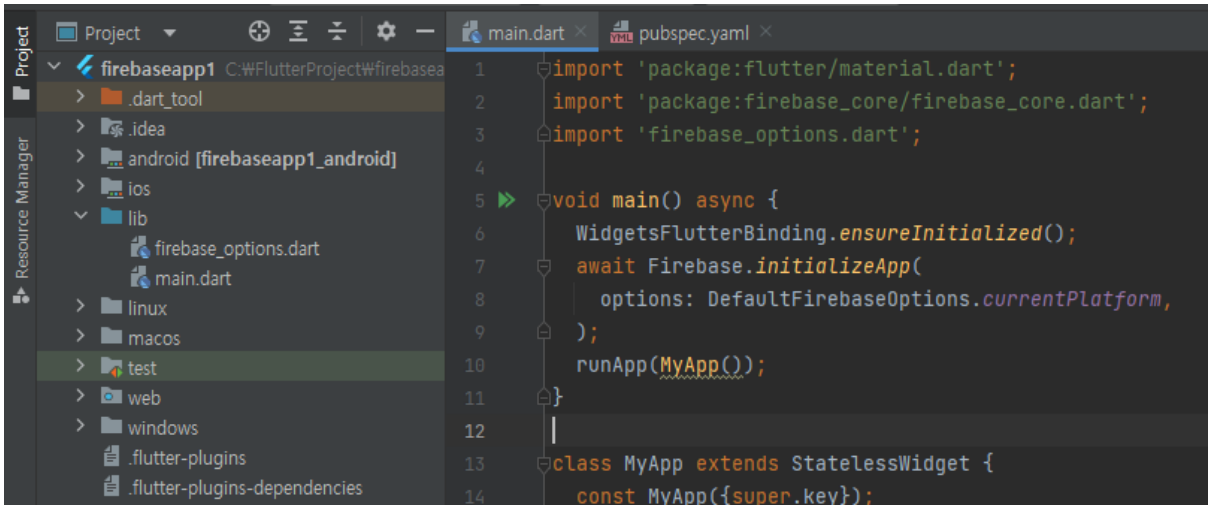
- **시스템환경변수 등록 후 시스템재시작**
- **플러터 프로젝트 디렉토리에서**
- **firebase logout**
- **firebase login**
- **dart pub global activate flutterfire_cli**
- **flutterfire configure --project=flutterlab-xxxxx**

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows


PS C:\Users\picbu> firebase logout
+  Logged out from picbuddy@gmail.com
PS C:\Users\picbu> firebase login
i  Firebase optionally collects CLI and Emulator Suite usage and error reporting informa
tion to help improve our products. Data is collected in accordance with Google's privacy
 policy (https://policies.google.com/privacy) and is not used to identify you.
? Allow Firebase to collect CLI and Emulator Suite usage and error reporting information
? (Y/n)
```

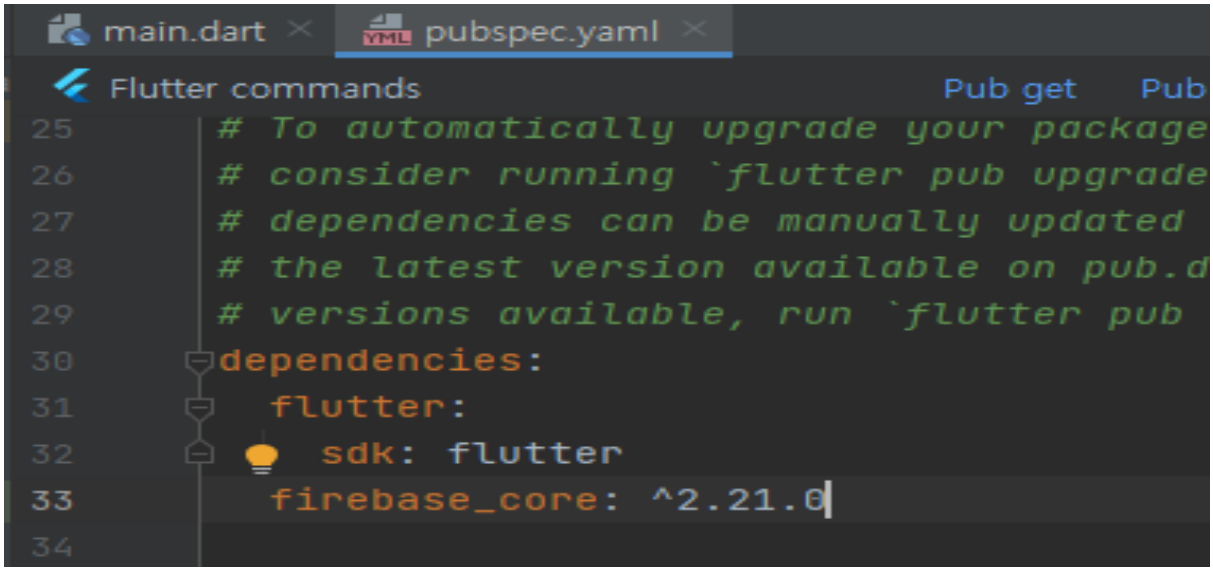


```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows

PS C:\Users\picbu> firebase logout
+  Logged out from picbuddy@gmail.com
PS C:\Users\picbu> firebase login
i  Firebase optionally collects CLI and Emulator Suite usage and error reporting information to help improve our products. Data
 is collected in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to identify you.

? Allow Firebase to collect CLI and Emulator Suite usage and error reporting information? Allow Firebase to collect CLI and Emu
lator Suite usage and error reporting
information? Yes
i  To change your data collection preference at any time, run `firebase logout` and log in again.

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhgmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent.com&sc
ope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww.googleapis.co
m%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response_type=code&state=1010114296&redirect_uri
=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...

+  Success! Logged in as picbuddy@gmail.com
PS C:\Users\picbu> cd C:\FlutterProject\firebaseapp1
PS C:\FlutterProject\firebaseapp1> dart pub global activate flutterfire_cli
Package flutterfire_cli is currently active at version 0.2.7.
The package flutterfire_cli is already activated at newest available version.
To recompile executables, first run `dart pub global deactivate flutterfire_cli`.
Installed executable flutterfire.
Activated flutterfire_cli 0.2.7.
PS C:\FlutterProject\firebaseapp1> flutterfire configure --project=flutterlab-8e978
i Found 9 Firebase projects. Selecting project flutterlab-8e978.
? Which platforms should your configuration support (use arrow keys & space to sel✓ Which platforms should your configuration s
upport (use arrow keys & space to select)? · android
i Firebase android app com.basic.firebaseapp1 registered.

Firebase configuration file lib\firebase_options.dart generated successfully with the following Firebase apps:

Platform  Firebase App Id
android   1:251021640366:android:fe133b1f98aadefe2be33f

Learn more about using this file and next steps from the documentation:
 > https://firebase.google.com/docs/flutter/setup
PS C:\FlutterProject\firebaseapp1>
```

Project coding

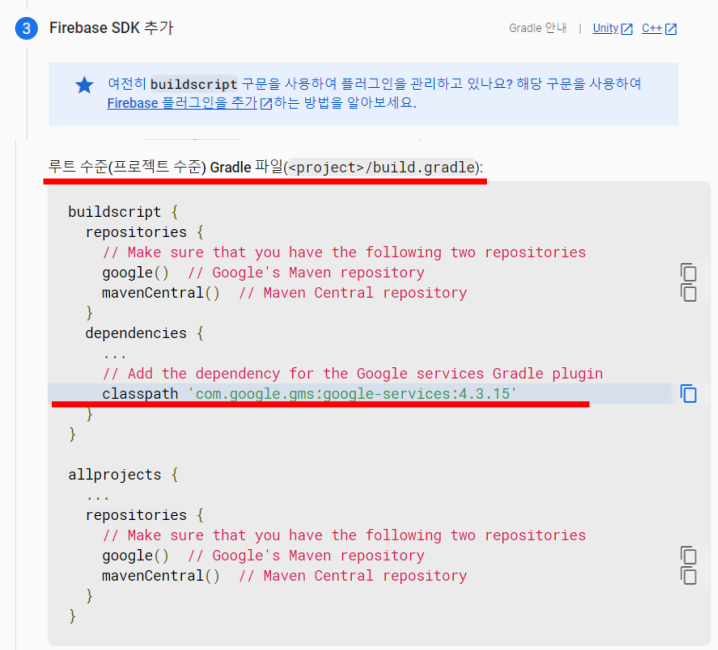


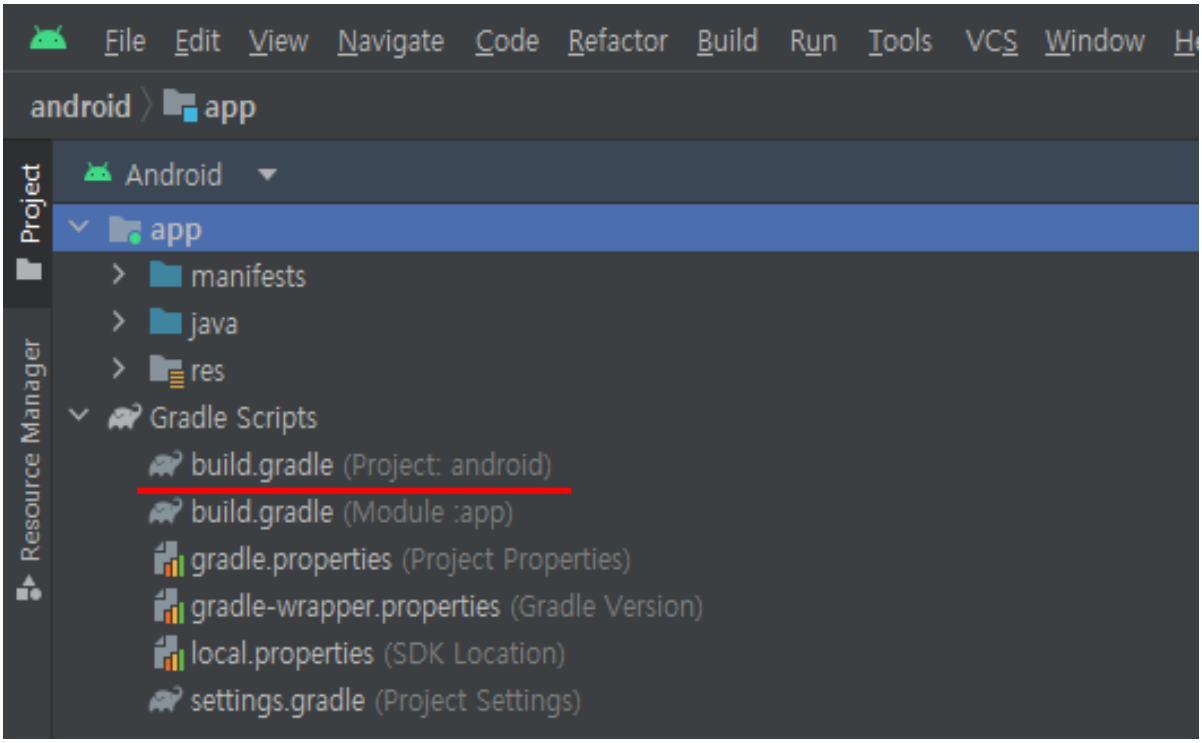
```dart
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(MyApp());
}
```



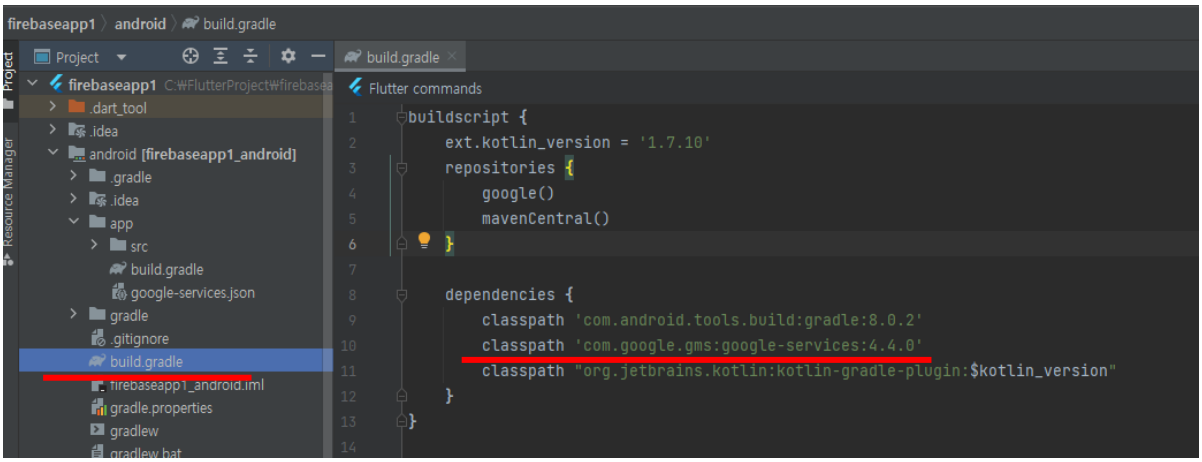
안드로이드 스튜디오 File → Open 메뉴로 android 폴더를 새 창으로 연다

→ **build.gradle(Project:android), build.gradle(Module:android.app) 에 services 를 등록한다.**



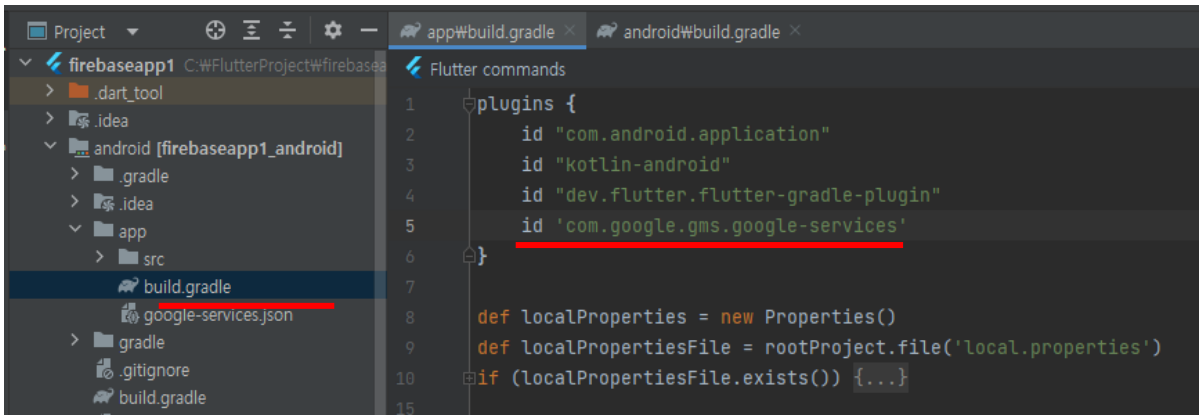


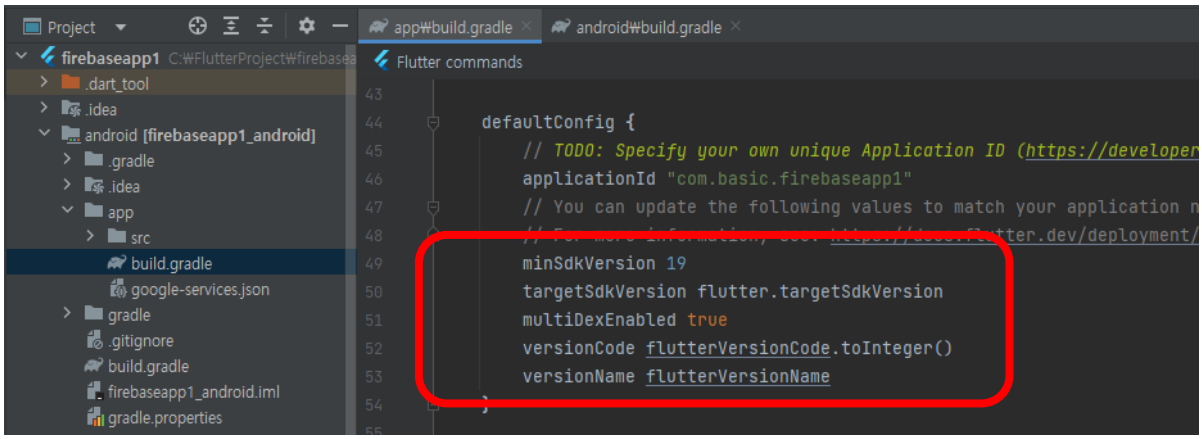
android₩build.gradle




```
dependencies {
    classpath 'com.android.tools.build:gradle:8.0.2'
    classpath 'com.google.gms:google-services:4.4.0'
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
}
```

app₩build.gradle





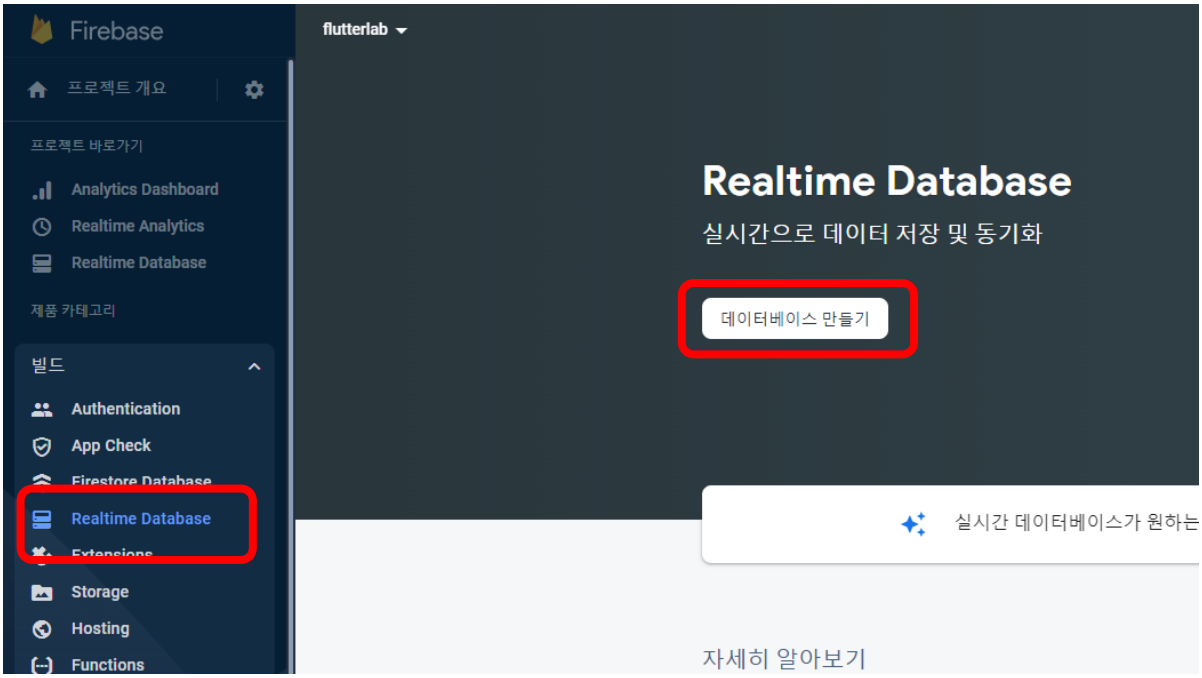

```
plugins {
    id "com.android.application"
    id "kotlin-android"
    id "dev.flutter.flutter-gradle-plugin"
    id 'com.google.gms.google-services'
}
```

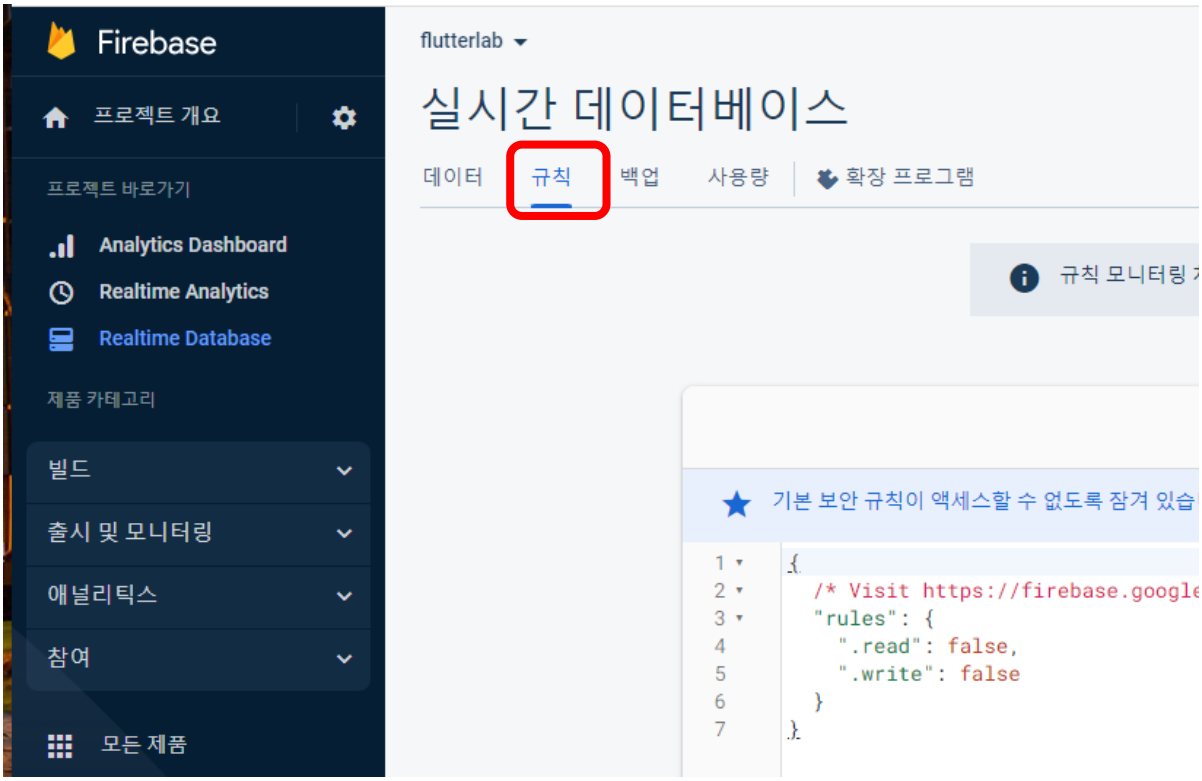
```
minSdkVersion 19
targetSdkVersion flutter.targetSdkVersion
multiDexEnabled true
```

Pubspec.yaml 파일에 firebase 추가

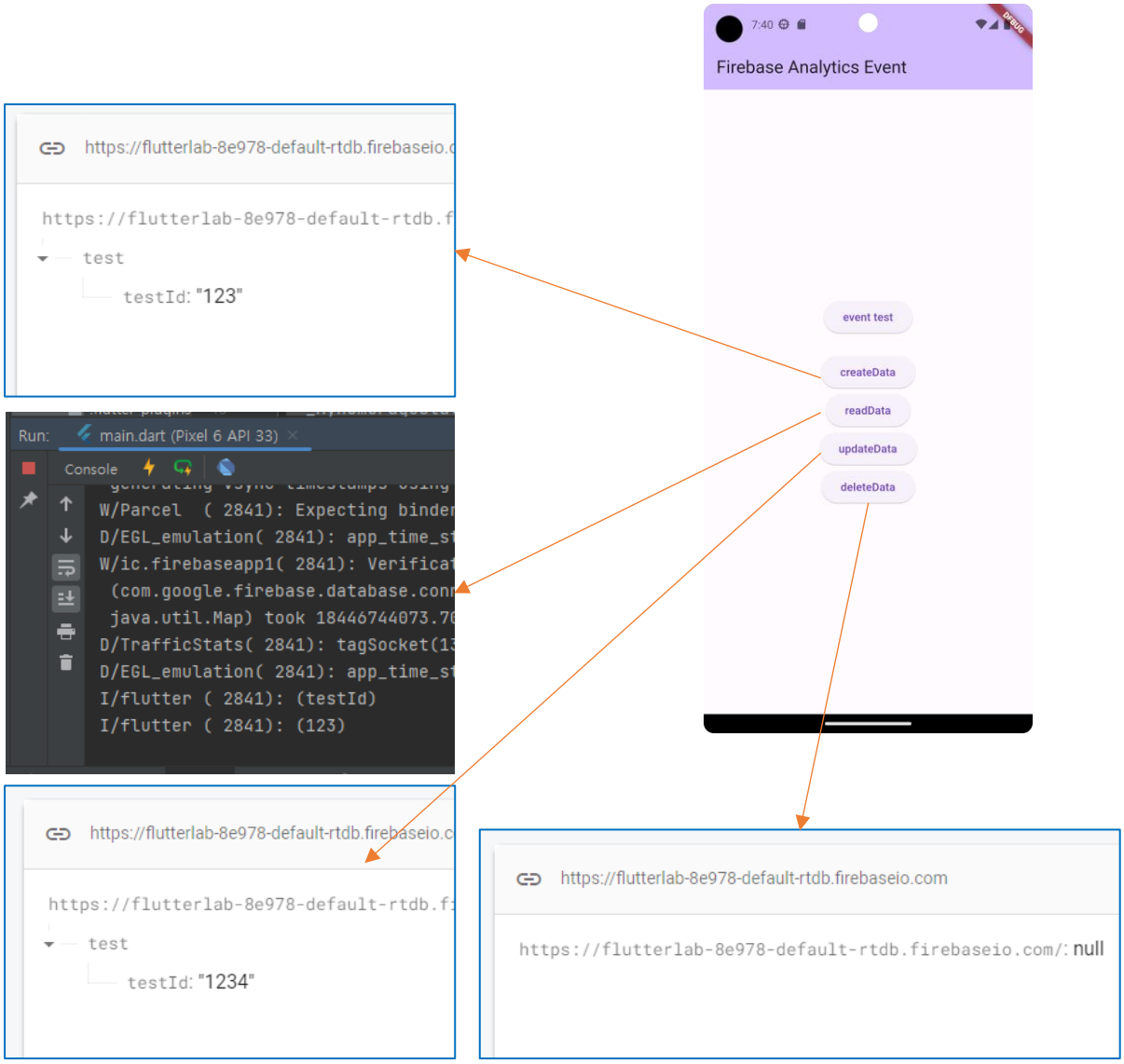
```yaml
dependencies:
  flutter:
    sdk: flutter
  firebase_core: ^2.21.0
  firebase_analytics: ^10.6.3
  firebase_database: ^10.3.3
```

Firebase database – CRUD app

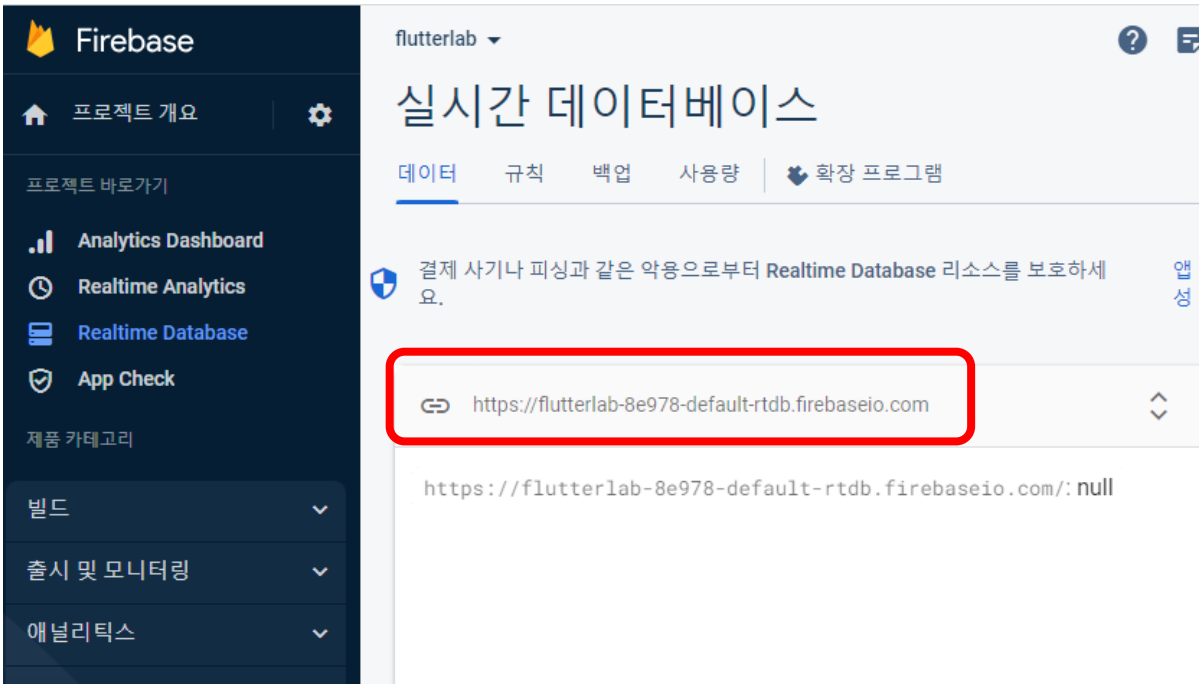


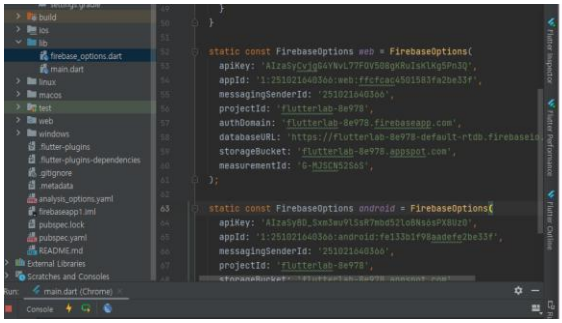

Firebase
프로젝트 개요
프로젝트 바로가기
Analytics Dashboard
Realtime Analytics
Realtime Database
제품 카테고리
빌드
출시 및 모니터링
애널리틱스
참여
모든 제품
flutterlab
실시간 데이터베이스
데이터
규칙
백업
사용량
확장 프로그램
규칙 모니터링
기본 보안 규칙이 액세스할 수 없도록 잠겨 있습
{
/* Visit https://firebase.google
"rules": {
".read": false,
".write": false
}
}

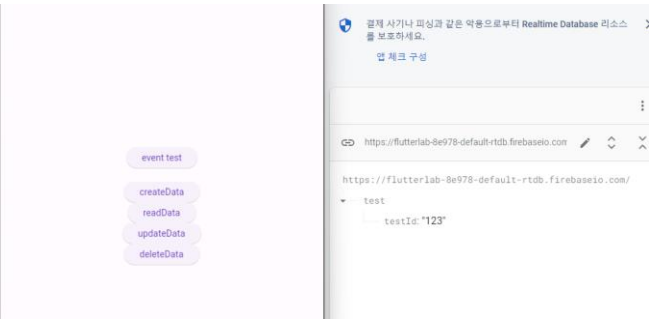

게시되지 않은 변경사항
게시
삭제
기본 보안 규칙이 액세스할 수 없도록 잠겨 있습니다.
{
/* Visit https://firebase.google.com/docs/databa
"rules": {
".read": true,
".write": true
}
}



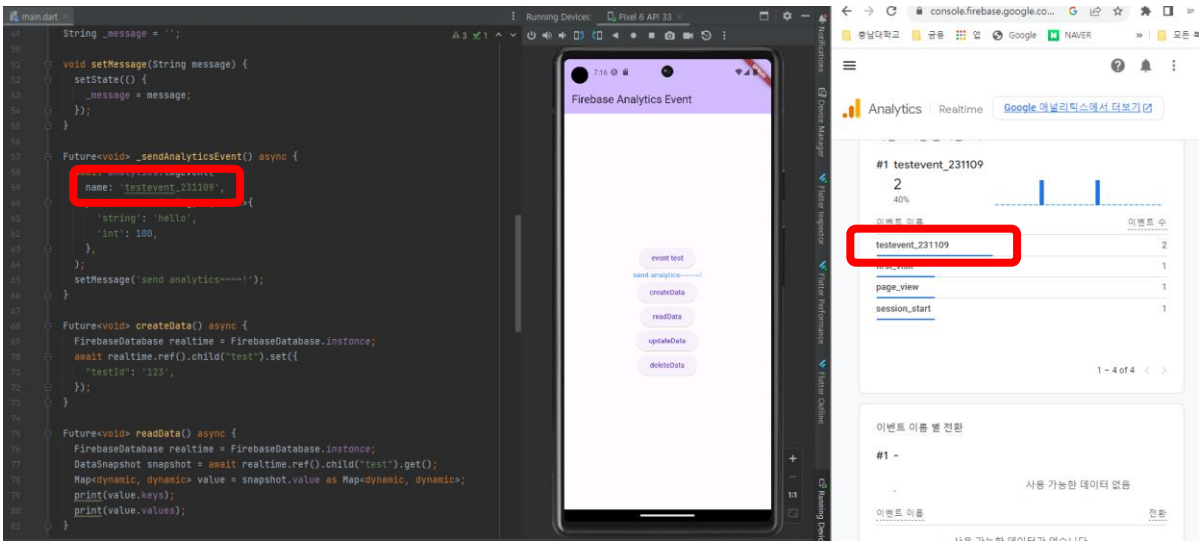
Web 으로 실행할때는 URL 필요!

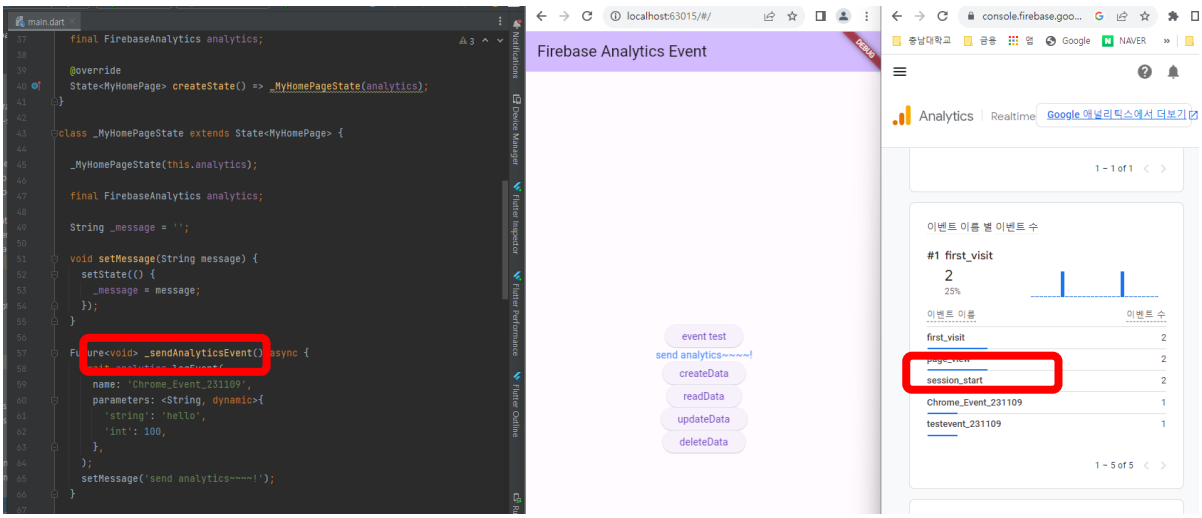

Firebase
프로젝트 개요
프로젝트 바로가기
Analytics Dashboard
Realtime Analytics
Realtime Database
App Check
제품 카테고리
빌드
출시 및 모니터링
애널리틱스
flutterlab
실시간 데이터베이스
데이터
규칙
백업
사용량
확장 프로그램
결제 사기나 피싱과 같은 악용으로부터 Realtime Database 리소스를 보호하세요.
https://flutterlab-8e978-default-rtdb.firebaseio.com
https://flutterlab-8e978-default-rtdb.firebaseio.com/: null




event test
createData
readData
updateData
deleteData

https://flutterlab-8e978-default-rtdb.firebaseio.com/
test
testId: "123"


main.dart
String _message = '';
void setMessage(String message) {
setState(() {
_message = message;
});
}
Future<void> _sendAnalyticsEvent() async {
name: 'testevent_231109',
'string': 'hello',
'int': 100,
setMessage('send analytics~~~~!');
}
Future<void> createData() async {
FirebaseDatabase realtime = FirebaseDatabase.instance;
await realtime.ref().child("test").set({
"testId": '123',
});
}
Future<void> readData() async {
FirebaseDatabase realtime = FirebaseDatabase.instance;
DataSnapshot snapshot = await realtime.ref().child("test").get();
Map<dynamic, dynamic> value = snapshot.value as Map<dynamic, dynamic>;
print(value.keys);
print(value.values);
}
Running Devices: Pixel 6 API 33
Firebase Analytics Event
event test
send analytics~~~~!
createData
readData
updateData
deleteData
console.firebase.google.co...
Analytics
Realtime
Google 애널리틱스에서 더보기
#1 testevent_231109
2
40%
이벤트 이름
이벤트 수
testevent_231109
2
page_view
1
session_start
1
1 – 4 of 4
이벤트 이름 별 전환
#1 -
사용 가능한 데이터 없음
이벤트 이름
전환


main.dart
final FirebaseAnalytics analytics;
@override
State<MyHomePage> createState() => _MyHomePageState(analytics);
class _MyHomePageState extends State<MyHomePage> {
_MyHomePageState(this.analytics);
final FirebaseAnalytics analytics;
String _message = '';
void setMessage(String message) {
setState(() {
_message = message;
});
}
name: 'Chrome_Event_231109',
parameters: <String, dynamic>{
'string': 'hello',
'int': 100,
setMessage('send analytics~~~~!');
}
localhost:63015/#/
Firebase Analytics Event
event test
send analytics~~~~!
createData
readData
updateData
deleteData
console.firebase.goo...
Analytics
Realtime
Google 애널리틱스에서 더보기
1 – 1 of 1
이벤트 이름 별 이벤트 수
#1 first_visit
2
25%
이벤트 이름
이벤트 수
first_visit
2
session_start
2
Chrome_Event_231109
1
testevent_231109
1
1 – 5 of 5

main.dart

```
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_analytics/firebase_analytics.dart';
import 'package:firebase_database/firebase_database.dart';
import 'firebase_options.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  static FirebaseAnalytics analytics = FirebaseAnalytics.instance;

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: MyHomePage(title: 'Firebase Analytics Event', analytics: analytics),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title, required this.analytics});
```

```
  final String title;
  final FirebaseAnalytics analytics;

  @override
  State<MyHomePage> createState() => _MyHomePageState(analytics);
}

class _MyHomePageState extends State<MyHomePage> {

  _MyHomePageState(this.analytics);

  final FirebaseAnalytics analytics;

  String _message = '';

  void setMessage(String message) {
    setState(() {
      _message = message;
    });
  }

  Future<void> _sendAnalyticsEvent() async {
    await analytics.logEvent(
      name: 'testevent_231109',
      parameters: <String, dynamic>{
        'string': 'hello',
        'int': 100,
      },
    );
    setMessage('send analytics~~~~!');
  }

  Future<void> createData() async {
    FirebaseDatabase realtime = FirebaseDatabase.instance;
    await realtime.ref().child("test").set({
      "testId": '123',
    });
```

```
}

Future<void> readData() async {
  FirebaseDatabase realtime = FirebaseDatabase.instance;
  DataSnapshot snapshot = await realtime.ref().child("test").get();
  Map<dynamic, dynamic> value = snapshot.value as Map<dynamic, dynamic>;
  print(value.keys);
  print(value.values);
}

Future<void> updateData() async {
  FirebaseDatabase realtime = FirebaseDatabase.instance;
  await realtime.ref().child("test").update({"testId": '1234'});
}

Future<void> deleteData() async {
  FirebaseDatabase realtime = FirebaseDatabase.instance;
  await realtime.ref().child("test").remove();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: Text(widget.title),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          ElevatedButton(
            onPressed: _sendAnalyticsEvent,
            child: const Text('event test'),
          ),
          Text(
            _message,
```

```
            style: const TextStyle(color: Colors.blueAccent),
          ),
          ElevatedButton(
            onPressed: createData,
            child: const Text('createData'),
          ),
          ElevatedButton(
            onPressed: readData,
            child: const Text('readData'),
          ),
          ElevatedButton(
            onPressed: updateData,
            child: const Text('updateData'),
          ),
          ElevatedButton(
            onPressed: deleteData,
            child: const Text('deleteData'),
          ),
        ],
      ),
    ),
  );
 }
}
```

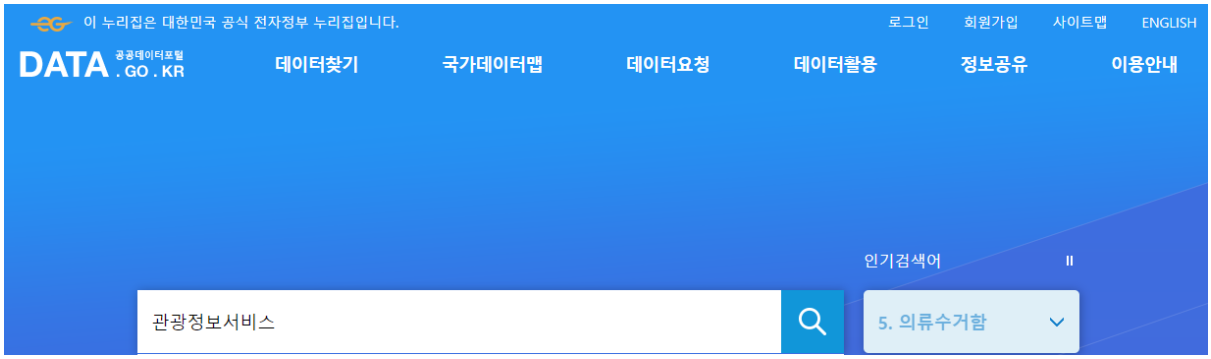
5-15. OpenAPI

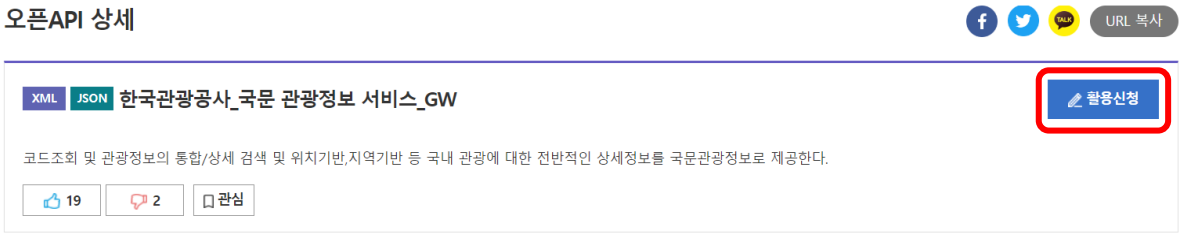
Open API - 관광정보

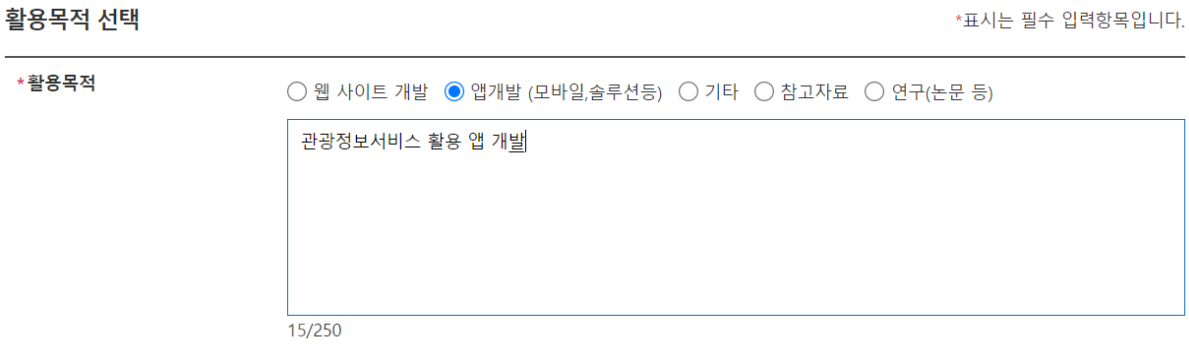
http://www.data.go.kr/ 에 일반 회원으로 가입

국문 관광 정보 서비스 검색 후 클릭

활용신청→활용목적 : (앱개발) 작성







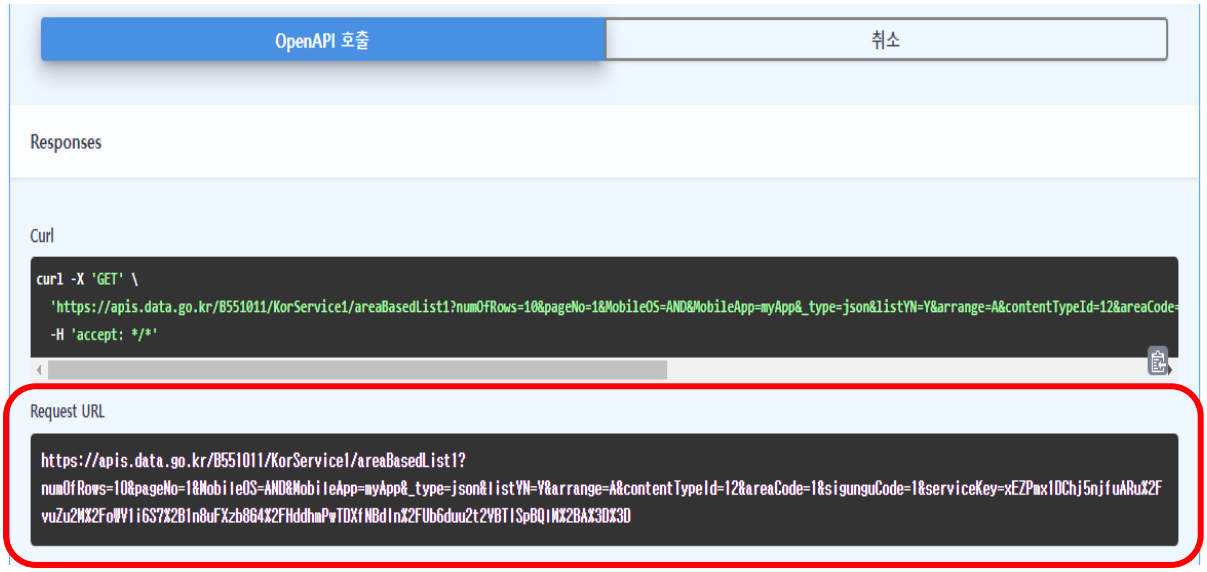
https://www.data.go.kr/data/15101578/openapi.do#tab_layer_detail_function

GET /areaBasedList1 지역기반 관광정보조회

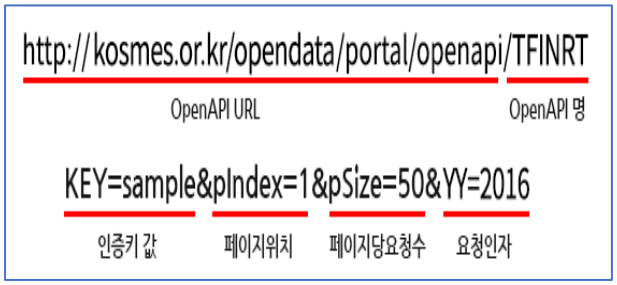
지역기반 관광정보파라미터 타입에 따라서 제목순,수정일순,등록일순 정렬검색목록을 조회하는 기능

Parameters 취소

Name	Description
numOfRows number (query)	한페이지결과수 10
pageNo number (query)	페이지번호 1
MobileOS * required string (query)	OS 구분 : IOS (아이폰), AND (안드로이드), WIN (윈도우폰), ETC(기타) AND
MobileApp * required string (query)	서비스명(어플명) myApp
_type string (query)	응답메세지 형식 : REST방식의 URL호출 시 json값 추가(디폴트 응답메세지 형식은XML) json
listYN string (query)	목록구분(Y=목록, N=개수) Y
arrange string (query)	정렬구분 (A=제목순, C=수정일순, D=생성일순) 대표이미지가반드시있는정렬(O=제목순, Q=수정일순, R=생성일순) A
contentTypeId string (query)	관광타입(12:관광지, 14:문화시설, 15:축제공연행사, 25:여행코스, 28:레포츠, 32:숙박, 38:쇼핑, 39:음식점) ID 12
areaCode string (query)	지역코드(지역코드조회 참고) 1
sigunguCode string (query)	시군구코드(지역코드조회 참고) 1
cat1 string (query)	대분류(서비스분류코드조회 참고) cat1
cat2 string (query)	중분류(서비스분류코드조회 참고) cat2
cat3 string (query)	소분류(서비스분류코드조회 참고) cat3
modifiedtime string (query)	수정일(형식 :YYYYMMDD) modifiedtime
serviceKey * required string (query)	인증키(서비스키) xEZPmx1DChj5njfuARu/vuZu2M/oWV1i6S7+1n8uf

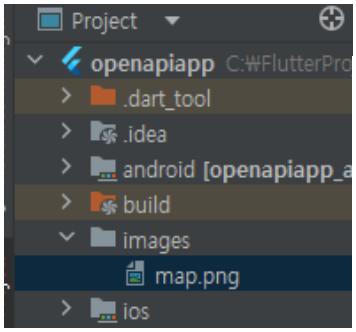


URL 형태



Response body(JSON)

pubspec.yaml



```
# To add assets to your application,
assets:
  - images/map.png
#   - images/a_dot_ham.jpeg
```

[실습]

tour.dart

```
class TourData {
  String? title;
  String? tel;
  String? zipcode;
  String? address;
  var id;
  var mapx;
  var mapy;
  String? imagePath;

  TourData(
  { this.id,
    this.title,
    this.tel,
    this.zipcode,
    this.address,
    this.mapx,
    this.mapy,
    this.imagePath
  });

  TourData.fromJason(Map data)
  : id = data['contentid'],
    title = data['title'],
    tel = data['tel'],
    zipcode = data['zipcode'],
    address = data['addr1'],
    mapx = data['mapx'],
    mapy = data['mapy'],
    imagePath = data['firstimage'];

  Map<String, dynamic> toMap() {
    return {
      'id' : id,
      'title' : title,
      'tel' : tel,
      'zipcode' : zipcode,
      'address' : address,
      'mapx' : mapx,
      'mapy' : mapy,
      'imagePath' : imagePath,
    };
  }
}
```

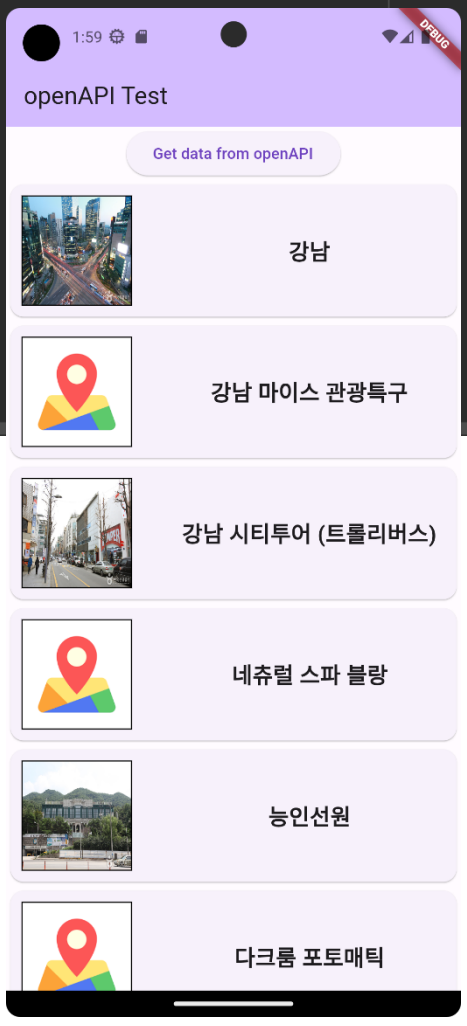
main.dart

```dart
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'tour.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'openAPI Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),  // ThemeData
      home: const MyHomePage(title: 'openAPI Test'),
    );  // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  String authKey =
      'xEZPmx1DChj5njfuARu%2FvuZu2M%2FoWV1i6S7%2B1n8uFXzb8G4%2FHddhmPwTDXfNBdIn%

  List<TourData> tourData = List.empty(growable: true);
```

```dart
List<TourData> tourData = List.empty(growable: true);

void getAreaList(
    {required int area,
    required int page,
    required int contentTypeId}) async {
  var url =
      'https://apis.data.go.kr/B551011/KorService1/'
      'areaBasedList1?numOfRows=10&pageNo=1&MobileOS=AND&MobileApp=myApp&'
      '_type=json&contentTypeId=12&areaCode=1&sigunguCode=1&'
      'serviceKey=xEZPmx1DChj5njfuARu%2FvuZu2M%2FoWV1i6S7%2B1n8uFXzb8G4%2FHddh

      /*'http://api.visitkorea.or.kr/openapi/service/rest/KorService/'
      'areaBasedList?ServiceKey=$authKey&MobileOS=AND&MobileApp=ModuTour&'
      '_type=json&areaCode=1&numOfRows=10&sigunguCode=$area&pageNo=$page&conte
      */
      /*'http://apis.data.go.kr/B551011/KorService1/'
      'areaBasedList1?ServiceKey=$authKey&MobileOS=AND&MobileApp=ModuTour&'
      '_type=json&areaCode=1&numOfRows=10&sigunguCode=$area&pageNo=$page&conte
      */

  var response = await http.get(Uri.parse(url));
  String body = utf8.decode(response.bodyBytes);
  //print(body);
  var json = jsonDecode(body);
  print(json);

  List jsonArray = json['response']['body']['items']['item'];

  tourData.clear();
  for (var s in jsonArray) {
    setState(() {
      tourData.add(TourData.fromJason(s));
    });
  }
}

```

```dart
ImageProvider<Object> getImage(String? imagePath) {
  if (imagePath != null && imagePath.length > 10) {
    print(imagePath);
    return NetworkImage(imagePath);
  } else {
    return const AssetImage('images/map.png');
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: Theme.of(context).colorScheme.inversePrimary,
      title: Text(widget.title),
    ),  // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          ElevatedButton(
              onPressed: () {
                getAreaList(area: 1, contentTypeId: 12, page: 1);
              },
              child: const Text('Get data from openAPI')),  // ElevatedButton
          Expanded(
            child: ListView.builder(
              itemBuilder: (context, index) {
                return Card(
                  child: InkWell(
                    child: Row(children: <Widget>[
                      Container(
                          margin: const EdgeInsets.all(10),
                          width: 100.0,
                          height: 100.0,
                          decoration: BoxDecoration(
                              shape: BoxShape.rectangle,
                              border:
                                  Border.all(color: Colors.black, width: 1),
                              image: DecorationImage(
```

```
                        decoration: BoxDecoration(
                            shape: BoxShape.rectangle,
                            border:
                                Border.all(color: Colors.black, width: 1),
                            image: DecorationImage(
                                fit: BoxFit.fill,
                                image:
                                    getImage(tourData[index].imagePath)))),  // Decora
                    const SizedBox(
                      width: 20,
                    ),  // SizedBox
                    Container(
                      width: MediaQuery.of(context).size.width - 150,
                      child: Column(
                        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                        children: <Widget>[
                          Text(
                            tourData[index].title!,
                            style: const TextStyle(
                                fontSize: 20,
                                fontWeight: FontWeight.bold),  // TextStyle
                          ),  // Text
                        ]),  // <Widget>[], Column
                    ),  // Container
                  ]),  // <Widget>[], Row
                ),  // InkWell
              );  // Card
            },
            itemCount: tourData.length,
          ),  // ListView.builder
        ),  // Expanded
      ],  // <Widget>[]
    ),  // Column
  ),  // Center
  );  // Scaffold
 }
}
```



결론

우리는 Design Thinking, Figma, Git, Dart, Flutter 를 공부하면서 모바일프로그래밍에 관련된 디자인개발방법론과 협업 툴 그리고 모바일 개발 플랫폼을 공부하였습니다. 실습 예제를 코딩하고, 이를 확장해 나가면서 자신의 코드로 만들기 바라며, 창의적이고 융합적인 사고를 통해 자신만의 프로젝트를 진행하여 모바일 프로그래밍 기술을 향상해 나가기 바랍니다.