



ITB Stikom  
Ambon

# PENGEMBANGAN APLIKASI ANDROID

## #4 Flutter Basic

By Chairil Ali, S.Kom



Di Modul ini, kita akan berkenalan dengan flutter basic, seperti bagaimana kita membuat project flutter, memahami struktur foldernya, struktur codenya dan juga akan membahas salah satu bagian dari flutter yaitu stateless widget dan stateful widget.

## **Membuat Project Flutter**

Dalam membuat project flutter ada 2 cara:

- 1.Menggunakan terminal/cmd,
- 2.Menggunakan IDE (VS Code, Android Studio)

## 1. Menggunakan terminal

Pertama buka dulu terminalnya, lalu ketikan perintah:

```
flutter create project_flutter_pertama
```

Ingat aturan dalam membuat nama project adalah:

- Semua huruf kecil
- Bila terdapat lebih dari 1 kata, dihubungkan dengan karakter underscore

```
(base) ✘ ~ flutter create project_flutter_pertama
Signing iOS app for device deployment using developer identity: "Apple
Development: saiful.bahri.tl@gmail.com (NX7PGXV2C4)"
Creating project project_flutter_pertama...
Running "flutter pub get" in project_flutter_pertama...                2,362ms
Wrote 127 files.

All done!
In order to run your application, type:

$ cd project_flutter_pertama
$ flutter run

Your application code is in project_flutter_pertama/lib/main.dart.

(base) ✘ ~ █
```

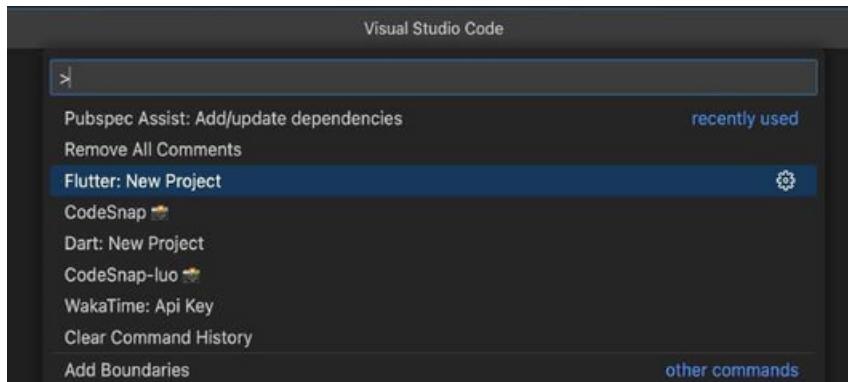
## 2. Menggunakan VS Code

Klik

- Windows : ctrl + shift + P
- Mac: command + shift + P

Lalu pilih, Flutter: New Project

Kemudian enter, pilih folder penyimpanan, lalu ketikan nama project flutter nya:  
project \_flutter\_pertama



## Membuka project flutter

Untuk cara pertama, untuk membuka project ke visual studio code, maka pertama kita perlu masuk dulu ke folder projectnya dengan perintah:

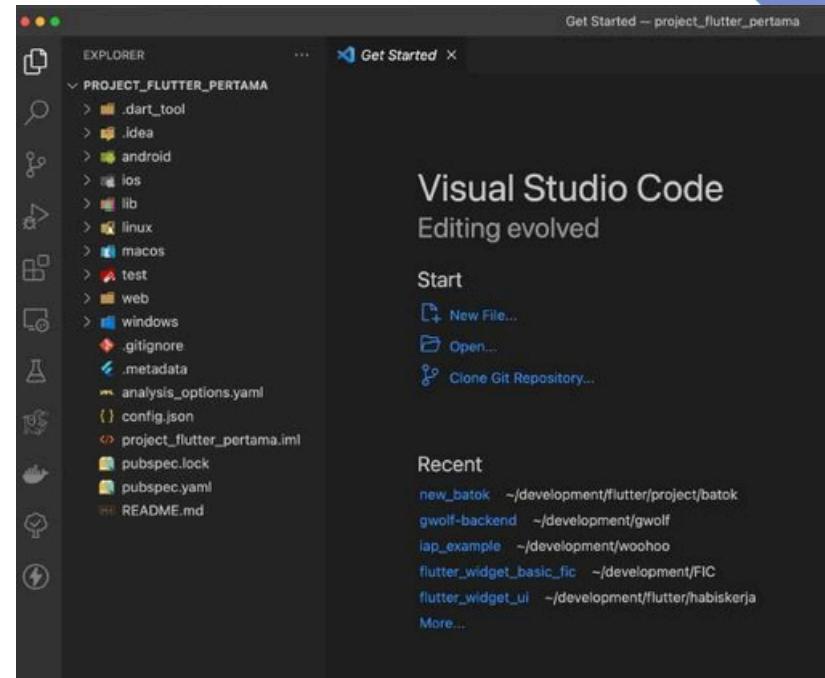
```
cd project_flutter_pertama
```

Setelah itu ketik perintah:

```
code .
```

Lalu enter

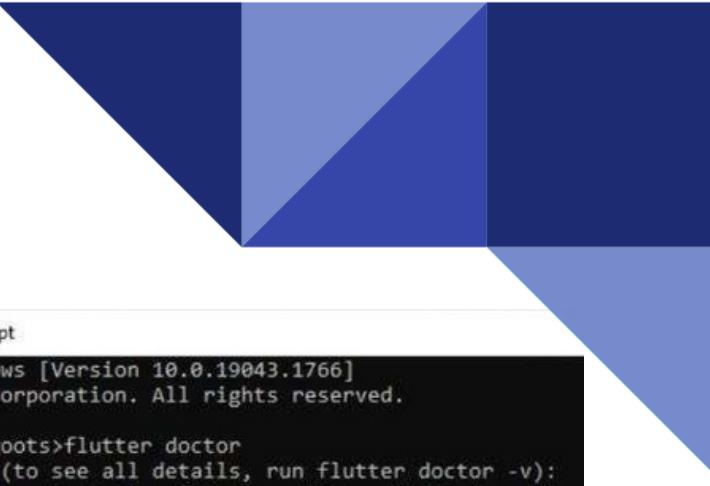
Untuk cara kedua, otomatis folder project flutter akan terbentuk dan dibuka oleh vscode



## Menjalankan Aplikasi Flutter

Untuk menjalankan aplikasi flutter di sistem operasi android menggunakan emulator, kita perlu memastikan dulu apakah android sdk sudah terinstall di laptop kita. Cara cek nya adalah dengan mengetikan perintah flutter doctor -v pada terminal.

Bila android toolchain belum centang hijau, ada beberapa langkah yang perlu dilakukan. Langkahnya sebagai berikut:



```
cmd Command Prompt
Microsoft Windows [Version 10.0.19043.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\bill_goots>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.0.5, on Microsoft Windows [Ver
[!] Android toolchain - develop for Android devices (Android
    X No Java Development Kit (JDK) found; You must have the
        your PATH. You can download the JDK from https://www.or
[✗] Chrome - develop for the web (Cannot find Chrome executab
    ! Cannot find Chrome. Try setting CHROME_EXECUTABLE to a
[✓] Visual Studio - develop for Windows (Visual Studio Build
[!] Android Studio (not installed)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

! Doctor found issues in 3 categories.

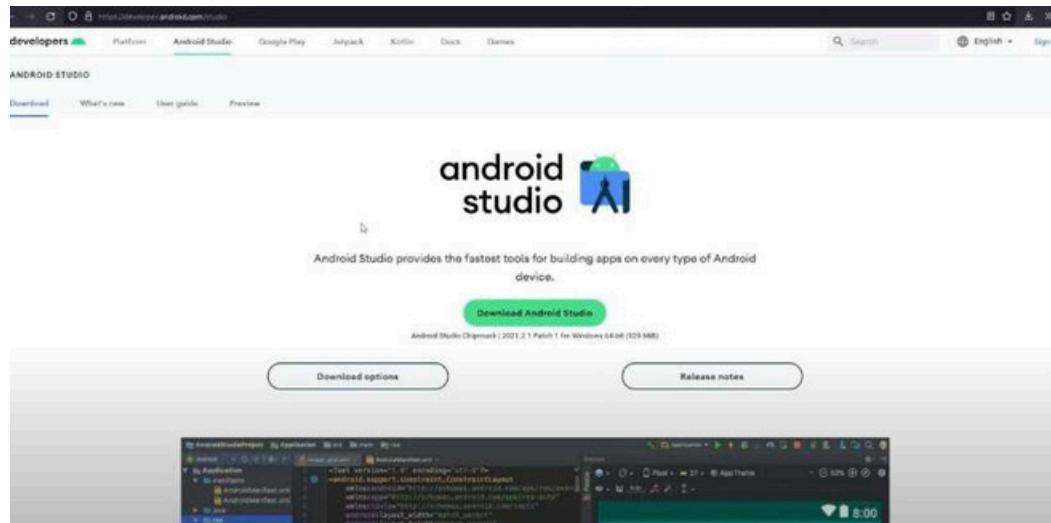
C:\Users\bill_goots>
```

# Menjalankan Aplikasi Flutter

## 1. Download Android Studio dan install

Software android studio bisa anda download dari link dibawah. Setelah selesai download, lalu install. Android studio disini akan menjadi tools untuk me manage android emulator.

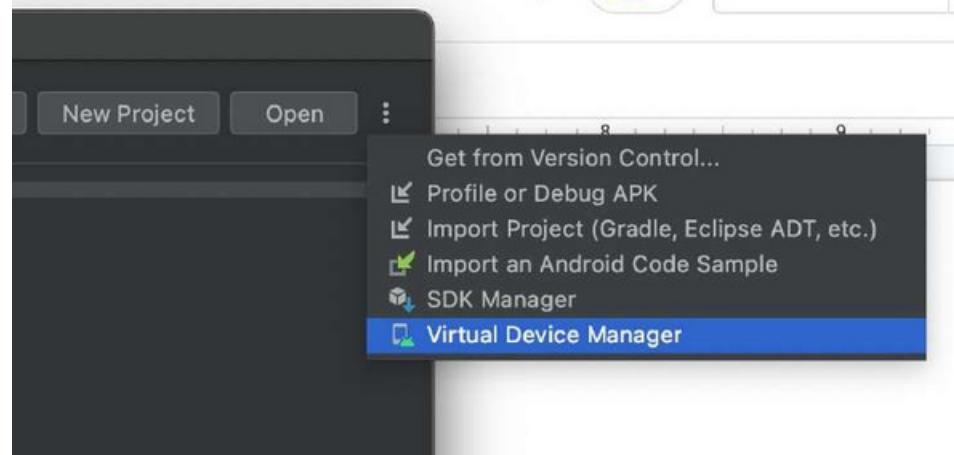
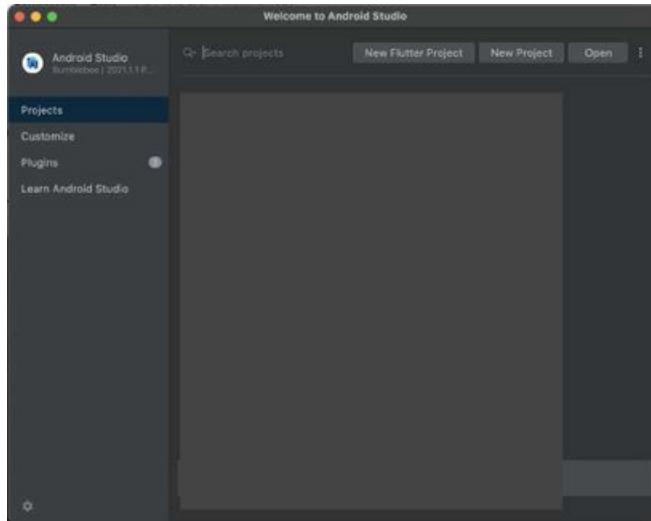
<https://developer.android.com/studio>



# Menjalankan Aplikasi Flutter

## 2. Buka Android studio

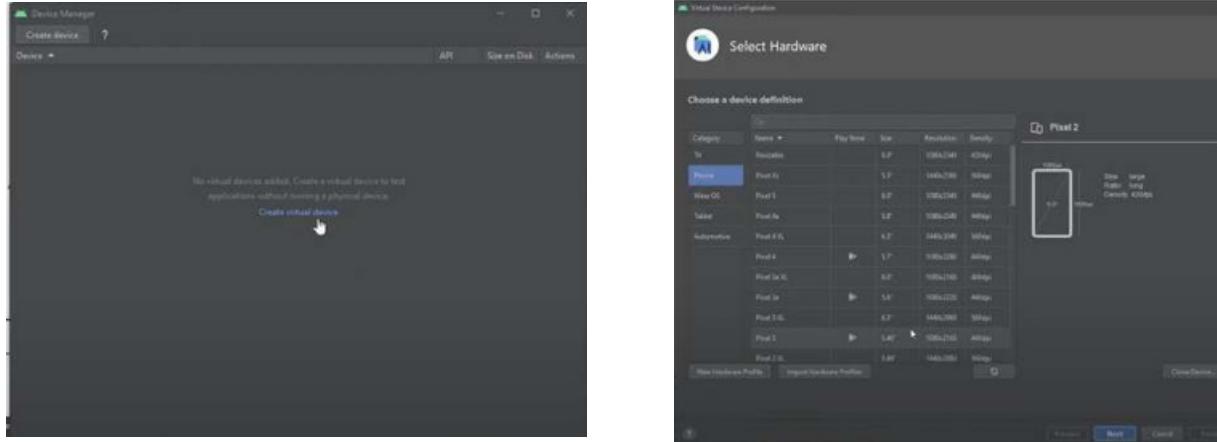
Setelah android studio dibuka, klik titik 3 lalu klik virtual device manager.



# Menjalankan Aplikasi Flutter

## 3. Pilih Device

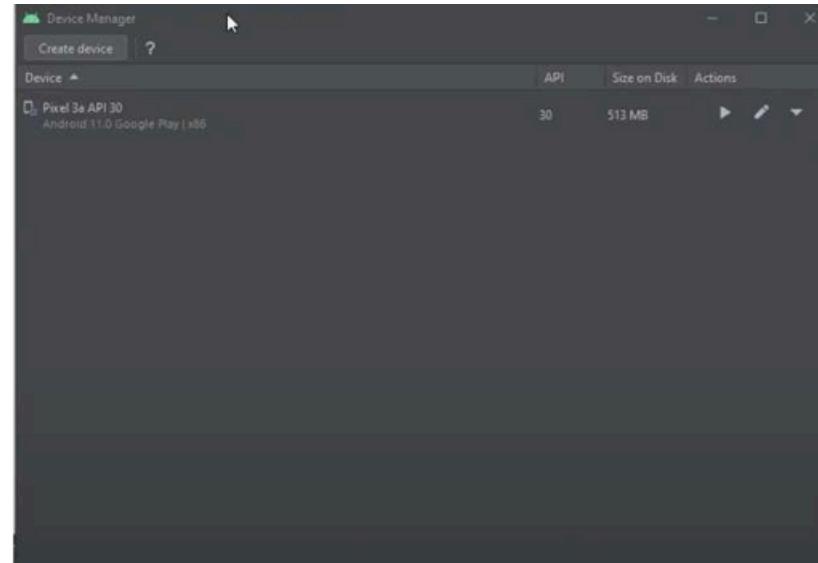
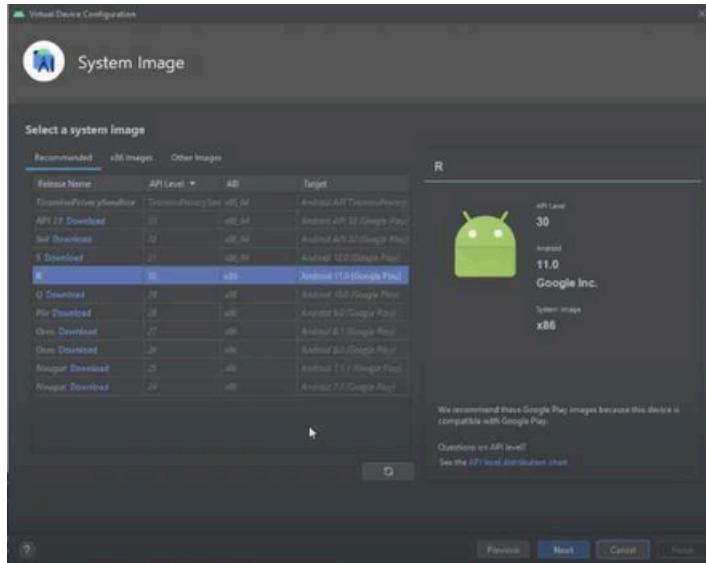
Selanjutnya kita diminta untuk memasukan tipe phone.



# Menjalankan Aplikasi Flutter

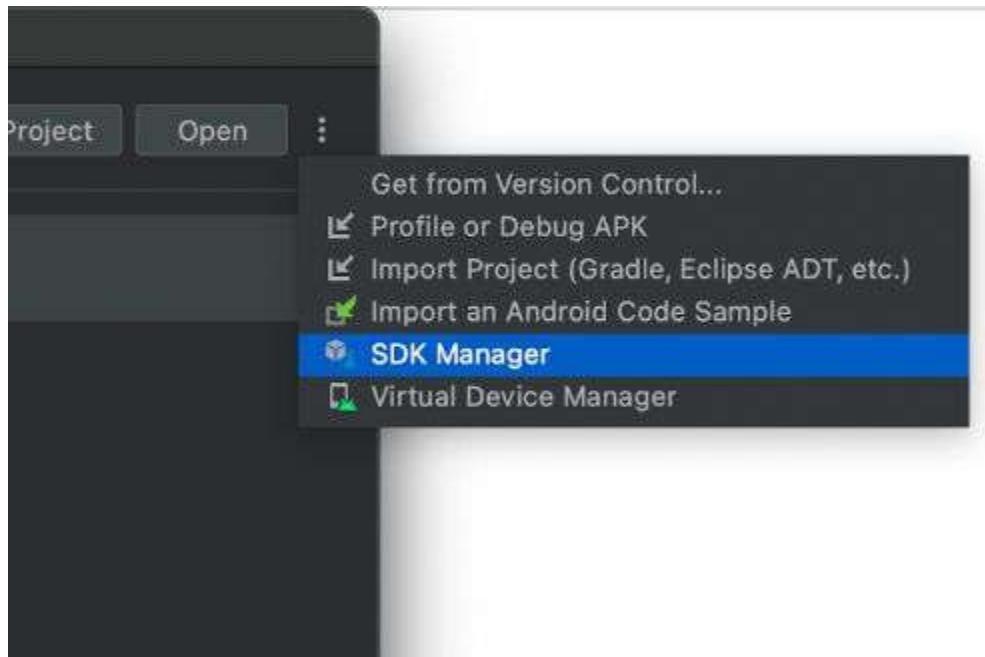
## 4. Pilih System Image

Setelah itu pilih system image Android. Download bila belum pernah download. Pastikan internet lancar karena akan download file yang besar. Next sampai Virtual Device terbentuk.



## Menjalankan Aplikasi Flutter

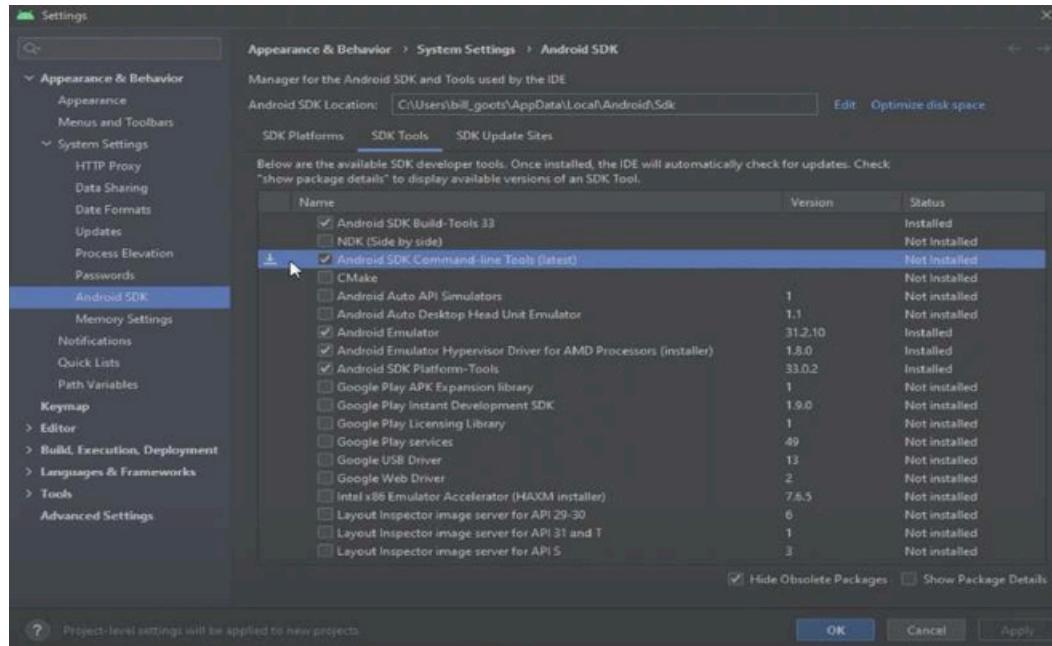
### 5. Masuk android studio dan pilih SDK Manager



# Menjalankan Aplikasi Flutter

## 6. Pilih Tab SDK Tools, centang Android SDK Command-line

Kemudian setelah berhasil download, kembali ke terminal kembali untuk check apakah android sdk nya siap untuk dikonsumsi.



# Menjalankan Aplikasi Flutter

## 7. flutter android --android-licenses (ketik y enter sampai selesai)

```
C:\Users\bill_goots>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[!] Flutter (Channel stable, 3.0.5, on Microsoft Windows [Version 10.0.19041.1052]
[!] Android toolchain - develop for Android devices (Android SDK version 30.0.3)
  X Android license status unknown.
    Run `flutter doctor --android-licenses` to accept the SDK licenses.
    See https://flutter.dev/docs/get-started/install/windows#android-setu
  X Chrome - develop for the web (Cannot find Chrome executable at .\GoogleCr
    ! Cannot find Chrome. Try setting CHROME_EXECUTABLE to a Chrome execut
[!] Visual Studio - develop for Windows (Visual Studio Build Tools 2019 16.10.31907.202)
[!] Android Studio (version 2021.2)
[!] VS Code (version 1.69.2)
[!] Connected device (2 available)
[!] HTTP Host Availability

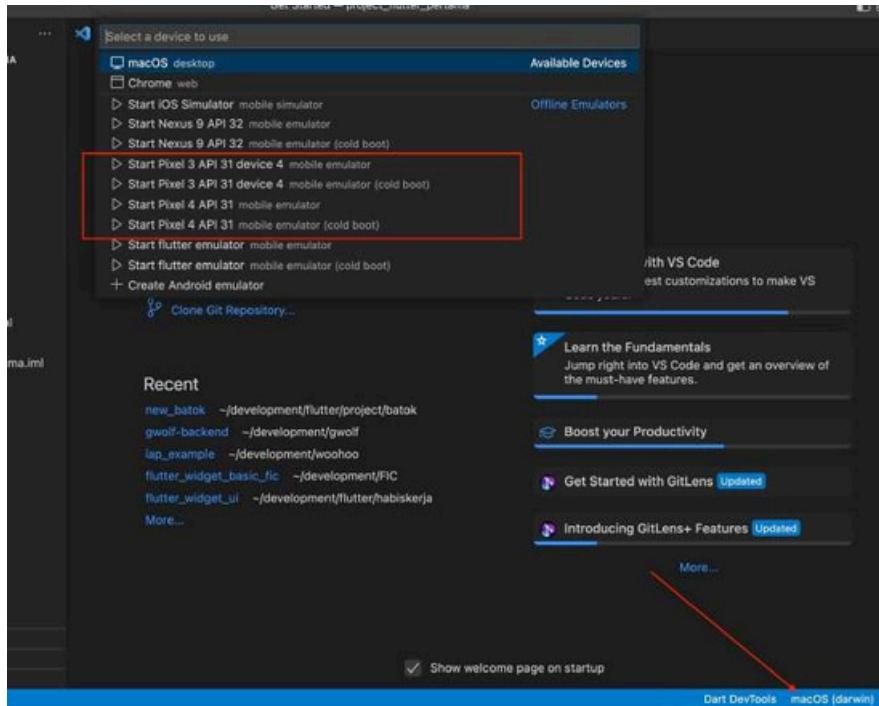
! Doctor found issues in 2 categories.

C:\Users\bill_goots>flutter doctor --android-licenses
Android sdkmanager not found. Update to the latest Android SDK and ensure
  resolve this 
C:\Users\bill_goots>
```

# Menjalankan Aplikasi Flutter

## Buka project sudah dibuat dengan vscode

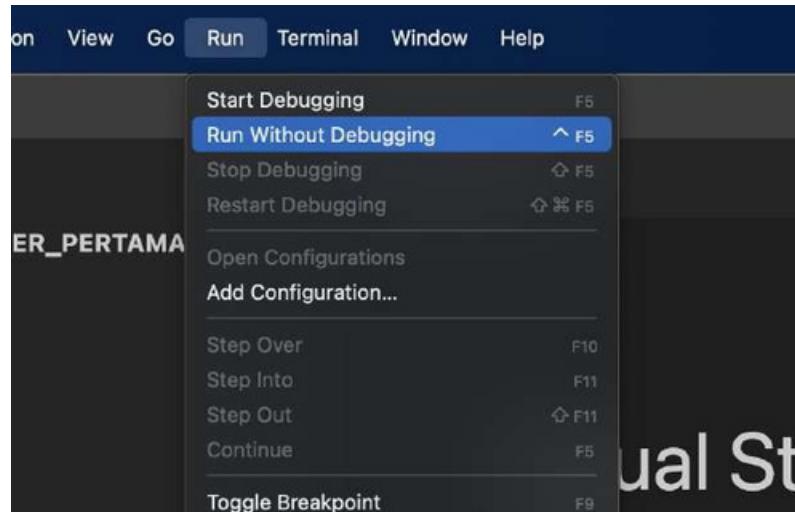
Disini kita akan melihat struktur folder dan pilihan device yang akan kita gunakan



# Menjalankan Aplikasi Flutter

## Pilih Device dan run aplikasi

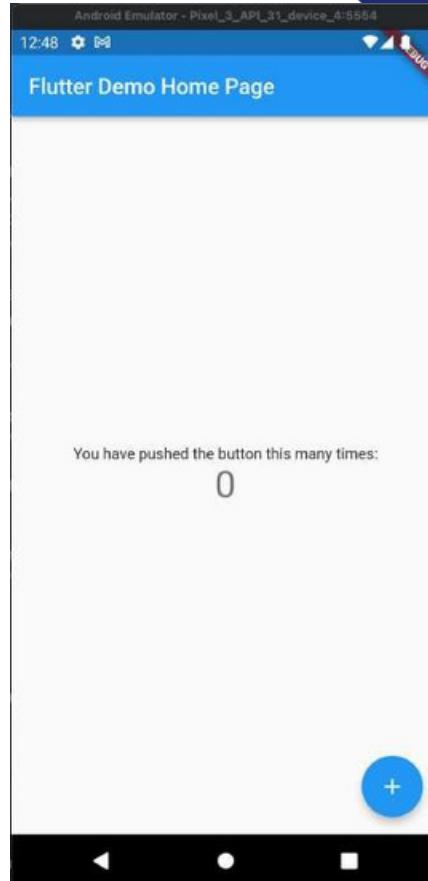
Setelah memilih device, kita dapat lanjut dengan cara klik menu Run dilanjut button run without debugging



## Menjalankan Aplikasi Flutter

### Aplikasi berhasil dijalankan

Selanjutnya bila tidak ada problem, maka aplikasi akan tampil di emulator yang sebelumnya sudah kita pilih.

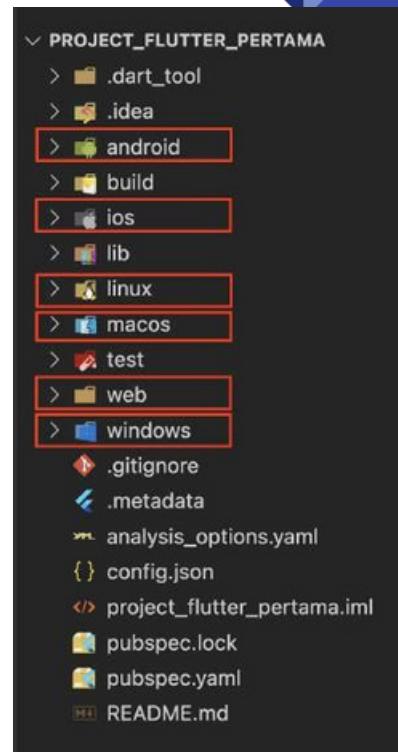


## Penjelasan file dan folder bawaan flutter

Setelah project terbentuk, kita akan melihat banyak folder dan file yang sudah dibuatkan oleh flutter. Yuk kita bahas file dan folder folder yang sudah di generate tersebut.

Folder, android, ios, linux, macos, web, dan windows adalah folder yang sudah disiapkan khusus oleh flutter supaya aplikasi yang kita buat bisa di jalankan di platform tersebut.

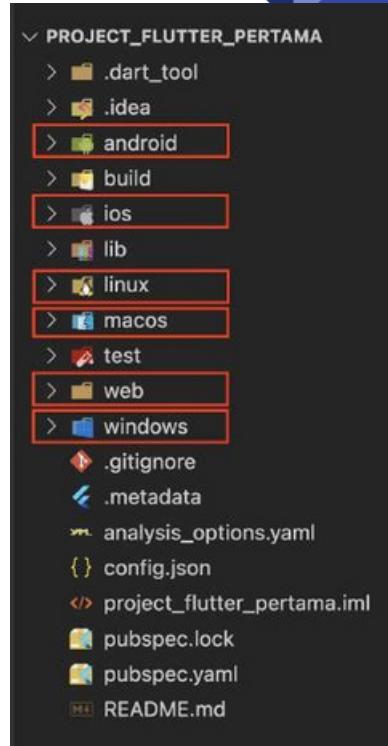
Disini nanti folder platform yang paling penting adalah folder android karena dalam case program flutter engineering kali ini kita hanya akan fokus pada development aplikasi android.



## Penjelasan file dan folder bawaan flutter

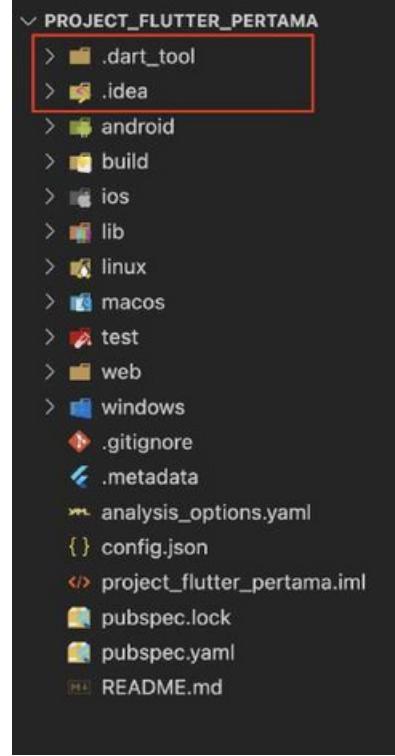
Folder android sendiri adalah folder yang penting karena didalamnya berisi project android lengkap dimana kita dapat membuatnya tanpa flutter, namun disini nanti flutter sdk akan menggabungkan kode flutter kita ke project android, jadi ketika kode flutter kita dikompilasi ke kode asli, pada dasarnya kode asli itu nanti akan disuntikkan ke dalam folder android untuk menjadi aplikasi asli android.

Dan untuk itu kita tidak begitu perlu merubah apapun di folder ini, dan sangat jarang sekali kita merubah kecuali ada beberapa penyesuaian konfigurasi yang berkaitan dengan update version dan sebagainya. Ini akan dibahas di materi selanjutnya.



Folder .idea ini menyimpan beberapa konfigurasi untuk android studio. Karena kita nantinya tidak menggunakan android studio untuk editornya, maka kita tidak butuh untuk mengubah apapun di dalam folder ini.

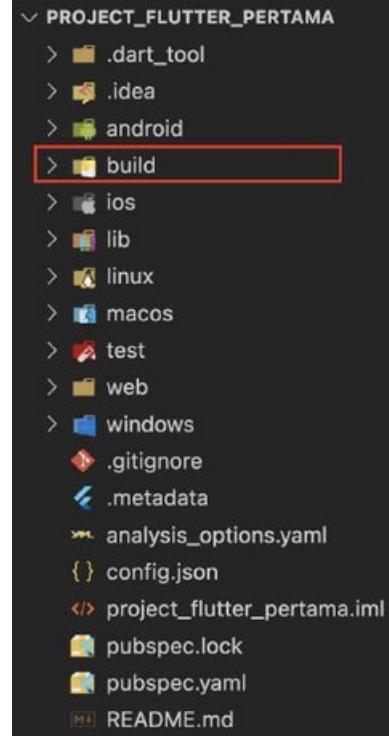
Folder .dart\_tool sendiri berisi konfigurasi dart package yang di generate oleh flutter. Folder ini pun tidak perlu kita ubah-ubah. Dapat kita abaikan terlebih dahulu.



## Penjelasan file dan folder bawaan flutter

Folder build ini akan penting karena folder ini nantinya akan menampung output dari aplikasi flutter kita.

Contoh file .apk, .abb, .ipa, .exe hasil build ke tiap platform nanti akan berada di folder ini.

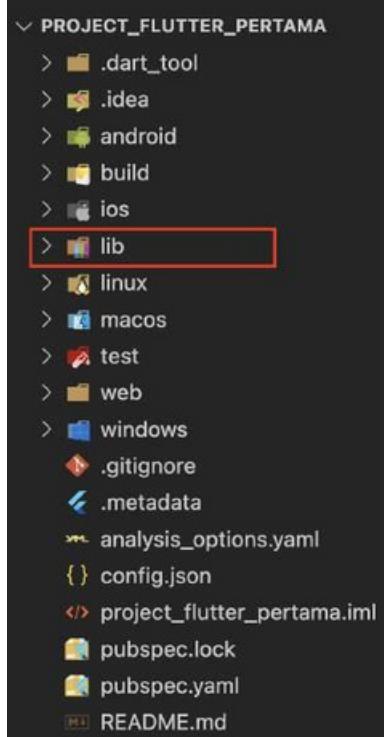


## Penjelasan file dan folder bawaan flutter

Folder lib ini sangat penting bagi kita karena 99% pekerjaan kita nantinya akan dilakukan di folder ini.

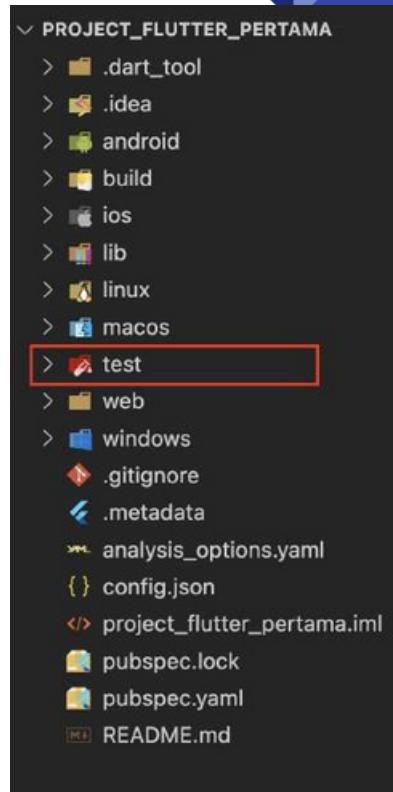
Lib sendiri adalah singkatan dari library dan itu artinya kode dart dan flutter yang kita tulis di dalam lib akan diartikan sebagai library yang akan disuntikan ke dalam kode asli tiap tiap platform.

Jadi folder ini adalah tempat kita menambah semua file dart dan tempat kita menulis kode.



## Penjelasan file dan folder bawaan flutter

Folder test untuk saat ini belum terlalu penting, didalam folder ini kita dapat menuliskan test untuk kode kita, yang nantinya dapat dijalankan secara otomatis. Salah satu fungsinya adalah ketika kita ada perubahan yang berkaitan kode yang pernah kita tulis test nya, dan ada impact, maka unit test ini akan memastikan apakah impact itu bermasalah apa tidak. Ini tentu penting ketika kita mengembangkan aplikasi yang sudah terlalu kompleks.



## Penjelasan file dan folder bawaan flutter

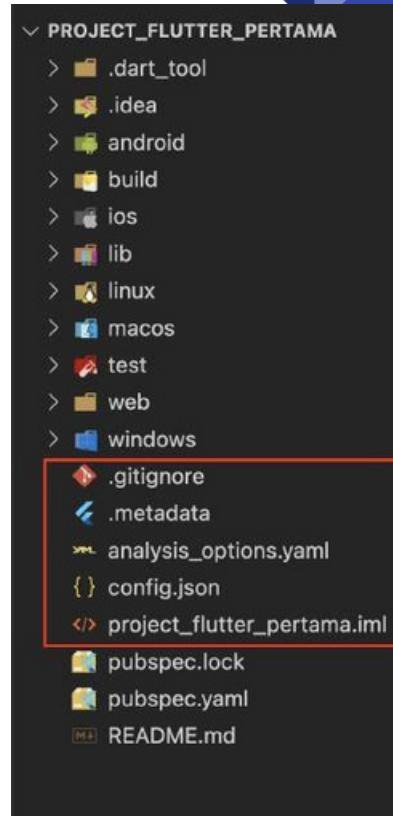
.gitignore berisi list folder atau file yang tidak akan ikut masuk kedalam git repository ketika kita push ke repository tersebut.

.metadata file ini untuk melacak properties di dalam project flutter, ini berfungsi sebagai alat flutter dalam menilai kemampuan dan peningkatan dan sebagainya, file ini tidak perlu diapa-apakan.

.analysis\_options.yaml berisi konfigurasi untuk melihat statistik analisis kode dart untuk mengecek error, warning, dan linter. Tidak perlu kita apa-apa kan

Config.json berisi konfigurasi tambahan ketika kita ingin run aplikasi untuk pertama kalinya di dalam proses development.

Project\_flutter\_pertama.iml file yang dikelola oleh flutter sdk dalam mengelola dipedensi internal dan beberapa pengaturan project. Ini file yang tidak perlu kita ubah.



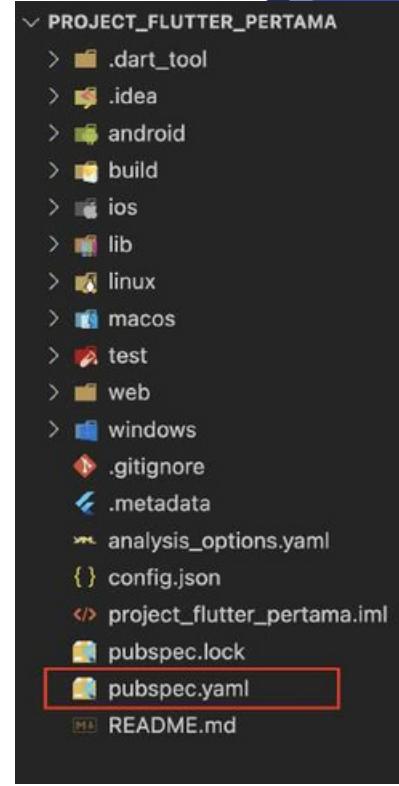
## Penjelasan file dan folder bawaan flutter

Pubspec.yaml merupakan file yang akan sering kita gunakan. File ini yang memungkinkan kita untuk mengelola sebagian besar dependensi project kita.

Apa artinya ini?

Artinya anda dapat mengkonfigurasi package pihak ketiga kedalam project kita dan semua fiturnya dapat langsung kita pakai.

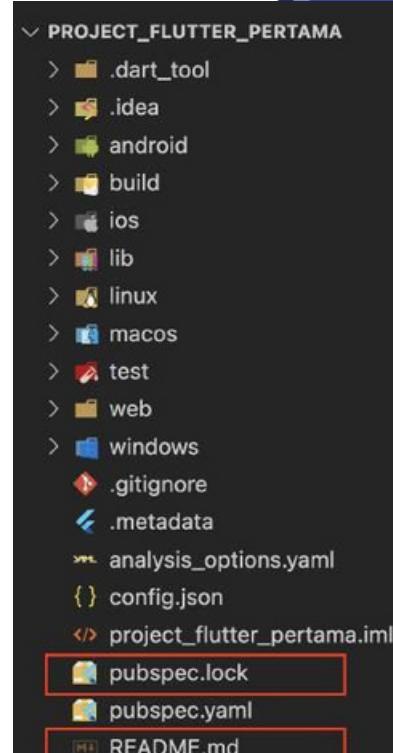
Kita juga dapat mengkonfigurasi beberapa hal lain disini, seperti misalnya font yang ingin kita pakai, atau gambar yang ingin kita import dari asset. Jadi file ini adalah file yang akan sering kita ubah untuk urusan konfigurasi project.



## Penjelasan file dan folder bawaan flutter

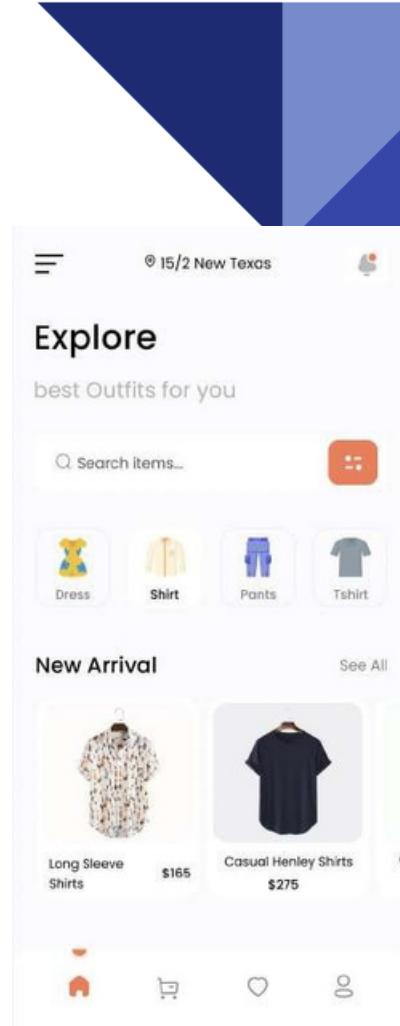
Pubspec.lock adalah file yang dihasilkan secara otomatis berdasarkan file .yaml, dan ini hanya menyimpan lebih banyak detail tentang semua dependensi yang kita miliki di project flutter kita. Namun ini file yang bukan kita kerjakan.

README.md ini bisa kita abaikan, bisa juga kita isi sebagai informasi tentang project yang sedang kita bangun. Jadi file ini dapat kita ubah sebagai informasi yang dapat berguna baik developer lain yang berhubungan dengan project flutter ini.



## Flutter Basic Widget

**Sebelum mulai membuat layout tampilan aplikasi, sebaiknya kita pahami dulu bagaimana flutter bekerja mengelola berbagai widget menjadi tampilan aplikasi yang tersusun rapi.**



## Flutter Basic Widget

Dalam project flutter, kita mempunya main function seperti halnya dart. function ini tidak memiliki kembalian dan tidak memerlukan argument.

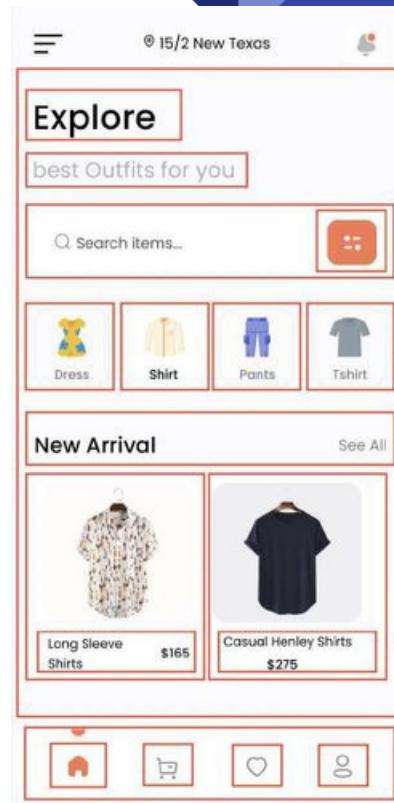
Seperti yang pernah kita pelajari, main adalah function yang secara otomatis dijalankan ketika aplikasi mulai dijalankan oleh flutter dan dart. Karena ada di file main.dart, kita tidak boleh mengganti namanya.

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Demo',
15     theme: ThemeData(
16       primarySwatch: Colors.blue,
17     ), // ThemeData
18     home: const MyHomePage(title: 'Flutter Demo Home Page'),
19   ); // MaterialApp
20 }
21 }
```

## Flutter Basic Widget

Untuk merubah kode menjadi tampilan dilayar, pertama kita perlu memahami, pada dasarnya tampilan dilayar itu adalah sekumpulan widget.

Flutter adalah tentang widget, dan setiap aplikasi yang kita buat hanya sekumpulan widget dan widget adalah blok penyusun UI yang dapat kita lihat di layar. Lihat yang saya kotakin merah, itu semua adalah widget. Jika disatukan di dalam satu widget, akan menjadi tampilan yang sesuai dengan kebutuhan UI.

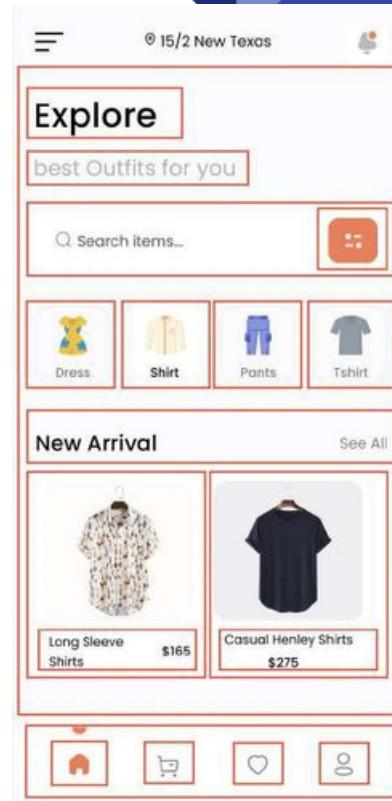


## Flutter Basic Widget

Widget juga sering berisi widget lain, seperti list daftar yang berisi daftar item. Seperti yang kita lihat di samping, dimana list navigation bar berisi widget item navigation bar.

Jadi sesungguhnya kita membuat aplikasi flutter kita ini menjadi widget tree, yang memiliki root widget dimana ini merupakan keseluruhan aplikasi kita. Sehingga apa

yang kita lihat di layar adalah widget yang menampung widget lain.



## Flutter Basic Widget

Karena semua yang ada di flutter adalah widget, yuk sekarang waktunya kita membuat widget kita sendiri.

Untuk membuat widget kita perlu membuat class widget adalah object, dan kita memerlukan class untuk membuat object.

Jadi mari lihat class MyApp baris 7 disamping. Ini adalah contoh class yang nantinya akan menjadi object widget.

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        primarySwatch: Colors.blue,
16      ), // ThemeData
17      home: const MyHomePage(title: 'Flutter Demo Home Page'),
18    ); // MaterialApp
19  }
20 }
21 }
```

## Flutter Basic Widget

Sekarang yang kita lihat adalah MyApp hanya menjadi sebuah class. Sedangkan widget yang benar-benar dapat kita lihat di layar bukanlah hal sepele untuk dibuat, karena pada akhirnya setiap pixel pada layar perlu kita kontrol.

Disinilah tugas flutter. Flutter sudah memiliki fungsi-fungsi tersebut dibalik layar, jadi kita tidak perlu menulisnya sendiri. Kita hanya perlu menggunakan fitur yang disebut pewarisan (inheritance).

Ini artinya kita akan membangun di atas class dasar, mendapatkan semua fitur dari class dasar tersebut dan hanya akan menambahkan fitur baru didalamnya.

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11   @override
12   Widget build(BuildContext context) {
13     return MaterialApp(
14       title: 'Flutter Demo',
15       theme: ThemeData(
16         primarySwatch: Colors.blue,
17       ), // ThemeData
18       home: const MyHomePage(title: 'Flutter Demo Home Page'),
19     ); // MaterialApp
20 }
21 }
```

## Flutter Basic Widget

Cara pewarisan class yang sudah dibuat oleh flutter adalah dengan menambahkan kata kunci extends setelah nama class dan sebelum kurung kurawal dan memberitahu dart bahwa class ini akan mewarisi class lain dan kita hanya dapat memperluas atau mewarisi satu class dalam satu waktu.

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11   @override
12   Widget build(BuildContext context) {
13     return MaterialApp(
14       title: 'Flutter Demo',
15       theme: ThemeData(
16         primarySwatch: Colors.blue,
17       ), // ThemeData
18       home: const MyHomePage(title: 'Flutter Demo Home Page'),
19     );
20   }
21 }
```

## Flutter Basic Widget

Terus bagaimana bisa kita menggunakan class dan library yang telah disediakan oleh flutter, jawabannya ada di pubspec.yaml.

Di file ini, kita dapat melihat bahwa dependencies pertama yang secara default dibuat ketika membuat project flutter adalah dependencies ke sdk flutter. Dimana sdk ini letak nya ada di folder lain di luar project ini.

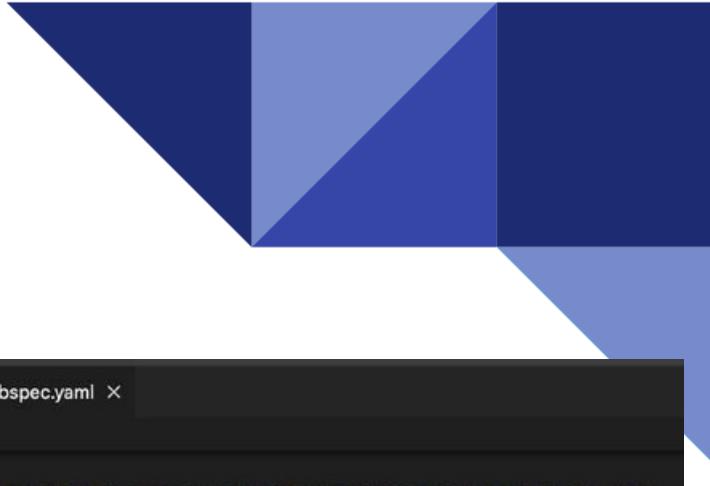


```
24
25  # Dependencies specify other packages that your package needs in order to work.
26  # To automatically upgrade your package dependencies to the latest versions
27  # consider running `flutter pub upgrade --major-versions`. Alternatively,
28  # dependencies can be manually updated by changing the version numbers below to
29  # the latest version available on pub.dev. To see which dependencies have newer
30  # versions available, run `flutter pub outdated`.
31  dependencies:
32    flutter:
33      |  sdk: flutter
34
35
```

## Flutter Basic Widget

Dengan adanya dependencies sdk flutter, kita dapat menggunakan semua class class nya kedalam project kita untuk kita perluas lagi dengan cara meng extends class class tersebut.

Di file ini pulalah nantinya kita dapat mendaftarkan package-package lain yang sudah disediakan oleh flutter ataupun yang dibuat oleh dev lain sudah di publish sebagai library yang dapat kita lihat di pub.dev  
Dalam penulisan package, perhatikan spasi, harus sejajar dengan packages flutter line 33.



```
main.dart pubspec.yaml X
pubspec.yaml

24
25 # Dependencies specify other packages that your package needs in order to work.
26 # To automatically upgrade your package dependencies to the latest versions
27 # consider running `flutter pub upgrade --major-versions`. Alternatively,
28 # dependencies can be manually updated by changing the version numbers below to
29 # the latest version available on pub.dev. To see which dependencies have newer
30 # versions available, run `flutter pub outdated`.
31 dependencies:
32   cupertino_icons: ^1.0.2
33   flutter:
34     sdk: flutter
35   image_picker: ^0.8.6
36
```

## Flutter Basic Widget

Setelah kita membuat class widget yang meng extends class stateless widget. Ada method yang wajib kita implement yaitu build(BuildContext context)

Method ini wajib mengembalikan widget karena disini kita bekerja dengan widget flutter, seluruh aplikasi kita adalah widget.

BuildContext sendiri adalah object khusus yang sudah disediakan oleh flutter lewat material.dart, dimana dia akan selalu membawa informasi meta tentang widget tersebut yang akan diteruskan ke build widget selanjutnya, dan informasi tersebut dapat digunakan oleh widget tree selanjutnya.

Dalam contoh diatas widget build mengembalikan widget MaterialApp()



```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11   @override
12   Widget build(BuildContext context) {
13     return MaterialApp(
14       title: 'Flutter Demo',
15       theme: ThemeData(
16         primarySwatch: Colors.blue,
17       ), // ThemeData
18       home: const MyHomePage(title: 'Flutter Demo Home Page'),
19     ); // MaterialApp
20   }
21 }
```

## Flutter Basic Widget

Class widget kita yang kita beri nama MyApp membutuhkan kembalian berupa widget juga. Dan disini saya akan menggunakan widget MaterialApp untuk kembalinya. MaterialApp ini adalah widget yang sudah disediakan oleh flutter melalui class material.dart

MaterialApp ini memiliki beberapa argumen yang dapat kita gunakan, janis argumennya adalah named argument. Seperti contoh disamping, saya menggunakan argument home untuk memasukan widget text yang bertuliskan Jago Flutter ke dalam widget MaterialApp yang nantinya akan diteruskan ke widget tree MyApp lalu di eksekusi dengan runApp dan akhirnya bisa tampil dilayar.

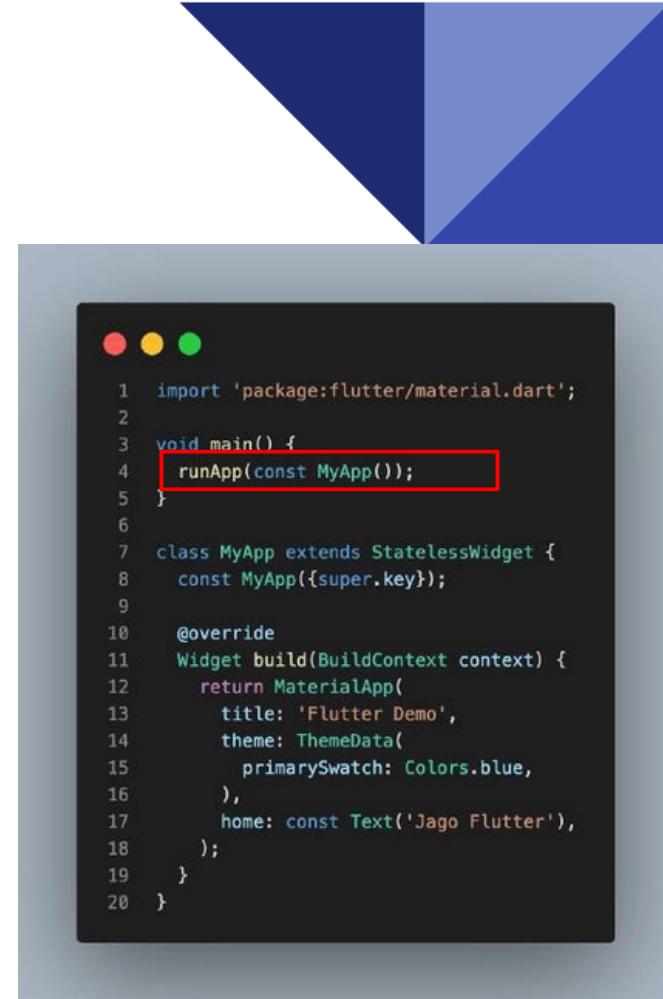
```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        primarySwatch: Colors.blue,
16      ),
17      home: const Text('Jago Flutter'),
18    );
19  }
20 }
```

## Flutter Basic Widget

Untuk menyambungkan widget yang kita buat sampai dengan tampil dilayar, kita akan butuh function yang sudah disediakan oleh flutter melalui material.dart nya yaitu runApp().

runApp sendiri function yang disediakan oleh flutter untuk menjalankan aplikasi flutter setelah aplikasi android atau ios di-boot. Alurnya dia akan mencoba mengambil widget tree yang kita buat, dan menggambarnya ke layar yang didasarkan pada widget tree tersebut. Jadi disini dia akan menggambar text Jago Flutter ke layar.

MyApp didalam runApp harus berupa function dengan cara memberinya kurung buka kurung tutup, karena tanpa itu, MyApp akan hanya menjadi tipe data.



```
1 import 'package:flutter/material.dart';
2
3 void main() {
4     runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8     const MyApp({super.key});
9
10    @override
11    Widget build(BuildContext context) {
12        return MaterialApp(
13            title: 'Flutter Demo',
14            theme: ThemeData(
15                primarySwatch: Colors.blue,
16            ),
17            home: const Text('Jago Flutter'),
18        );
19    }
20 }
```

## Flutter Basic Widget

Ketika di running aplikasi flutter kita, tampilannya akan seperti ini.

Dalam tahap ini kita baru memperlihatkan bagaimana text Jago Flutter yang sebelumnya berupa string, sekarang bisa tampil dilayar. Belum memperdulikan tampilan layar yang indah karena belum menambahkan widget lain selain widget Text().

Dari hasil yang kita lihat, ini membuktikan bahwa fungsi dasar berjalan baik. Dengan alur widget MyApp yang kita buat dengan tambahan fitur yang disediakan oleh widget StatelessWidget berupa method build() diterima oleh fungsi utama untuk menjalankan aplikasi dengan bantuan runApp.



## Flutter Basic Widget

Untuk mendapatkan tampilan yang lebih indah, kita dapat menggunakan widget lain yang sudah disediakan oleh flutter yaitu widget scaffold. Widget ini mempunyai beberapa argument, yang sering digunakan adalah argument appBar dan body.

Disamping saya contohkan untuk menambahkan widget scaffold yang saya masukan kedalam argument home yang ada di MaterialApp, lalu di dalam scaffold terdapat appBar. Ini yang nantinya akan menjadi tampilan di bagian atas layar. Lalu terdapat juga body, ini nanti berisi content yang ingin kita kelola.

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter Demo',
14      theme: ThemeData(
15        primarySwatch: Colors.blue,
16      ),
17      home: Scaffold(
18        appBar: AppBar(
19          title: const Text('Jago Flutter'),
20        ),
21        body: const Text('Belajar Bersama Jago Flutter'),
22      );
23    }
24 }
```

## Flutter Basic Widget

Hasil ketika kita jalankan akan seperti yang ada di sambil ini.

Terdapat appbar dengan title Jago Flutter, lalu untuk body nya sendiri masih berupa text dengan wording Belajar bersama jago flutter.

Setelah ini kita akan belajar mengenal widget, ada widget yang visible dan invisible. Maksudnya bagaimana yuk lanjut ke bab selanjutnya.

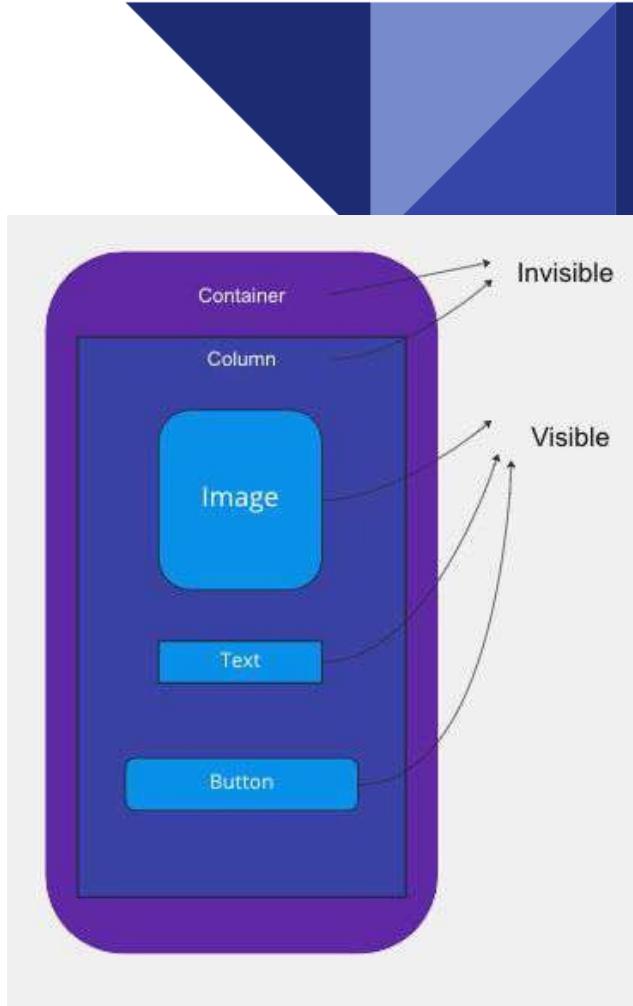


## Flutter Basic Widget

Di bab sebelumnya sudah mengenal widget text, dimana hasilnya itu dapat dilihat dalam berupa text di layar.

Disamping saya beri ilustrasi bahwa widget itu ada yang terlihat dan ada juga yang tidak terlihat. Baik yang terlihat maupun yang tidak terlihat kedua nya sama-sama penting.

Widget yang terlihat akan memberikan UI dan UX yang baik bagi user. User dapat berinteraksi dengan aplikasi. Untuk widget yang tidak nampak seperti listview, column, row, ini adalah widget-widget penting yang fungsinya untuk mengatur widget tree supaya widget yang nampak dapat disusun dengan rapi.



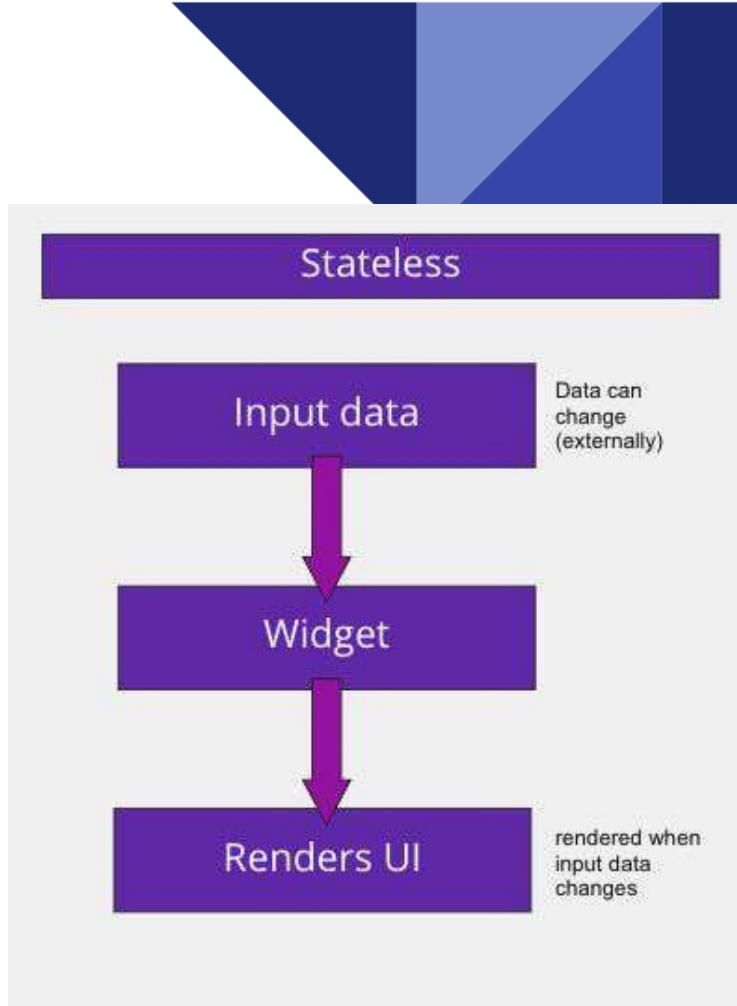
## Stateless Widget

Sebelum kita masuk ke stateless stateful pertama perlu tahu apa itu state ?

State adalah data atau informasi yang digunakan aplikasi atau widget dalam aplikasi kita.

State sendiri terbagi menjadi 2, yaitu app state dan local state. Stateless dan stateful ini masuk ke dalam local state.

Stateless sendiri widget yang tidak memiliki state, sehingga perubahan dan hasil render UI nya itu ditentukan oleh inputan yang masuk kedalam stateless widget tersebut.



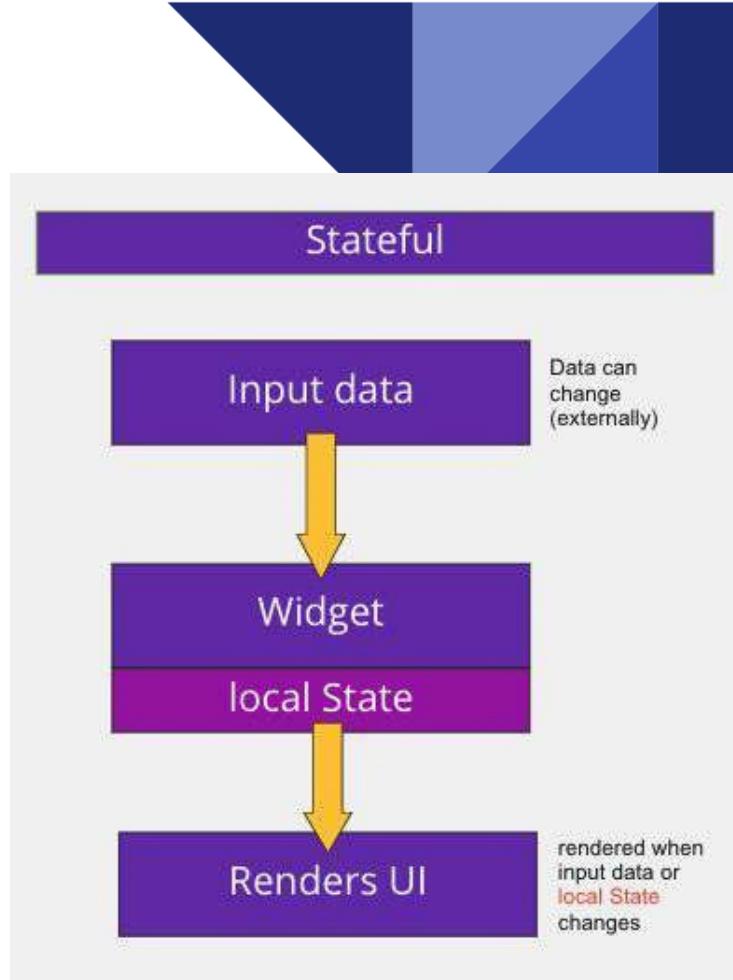
## Stateless Widget

Seperti kode yang kita lihat di samping ini. Class ShowTextWidget ini extends ke StatelessWidget yang artinya UI akan dirender ketika class ini dipanggil oleh widget lain dengan inputan berupa string text. Jadi dia tidak memiliki state sendiri di dalam dirinya.

```
1  class MyApp extends StatelessWidget {
2    const MyApp({super.key});
3
4    @override
5    Widget build(BuildContext context) {
6      return MaterialApp(
7        title: 'Flutter Demo',
8        theme: ThemeData(
9          primarySwatch: Colors.blue,
10        ),
11        home: Scaffold(
12          appBar: AppBar(
13            title: const Text('Jago Flutter'),
14          ),
15          body: const ShowTextWidget(
16            text: 'Belajar Bersama Jago Flutter',
17          ),
18        )));
19    }
20  }
21
22 class ShowTextWidget extends StatelessWidget {
23   final String text;
24   const ShowTextWidget({
25     Key? key,
26     required this.text,
27   }) : super(key: key);
28
29   @override
30   Widget build(BuildContext context) {
31     return Text(text);
32   }
33 }
```

## Stateful Widget

Untuk stateful dia adalah widget yang memiliki state didalamnya. Sehingga class yang extends class StatefulWidget, akan dapat memiliki internal state nya sendiri, hal ini dapat menguntungkan karena render UI nya tidak hanya bergantung dari inputan dari widget lain, namun dengan memanggil setState maka widget build akan rerun ulang dengan state yang baru tanpa harus menunggu perubahan di widget tree atas nya.



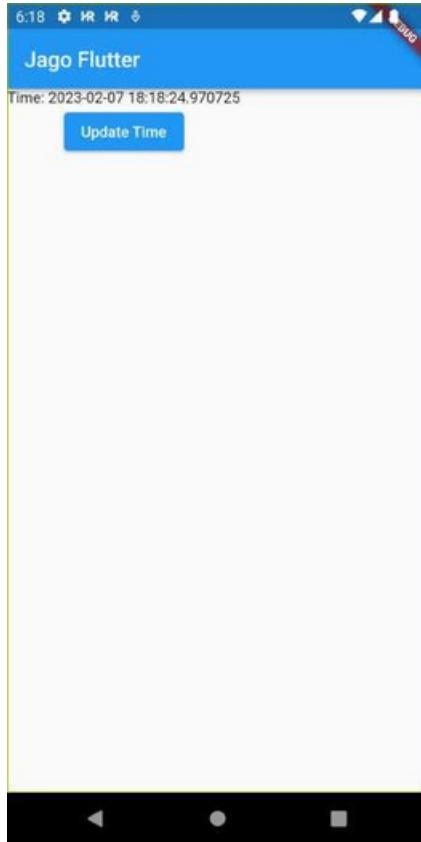
## Stateful Widget

Seperti class ChangeTextWidget di samping ini, dimana dia meng extends class StatefulWidget. Untuk susunan class nya seperti disamping.

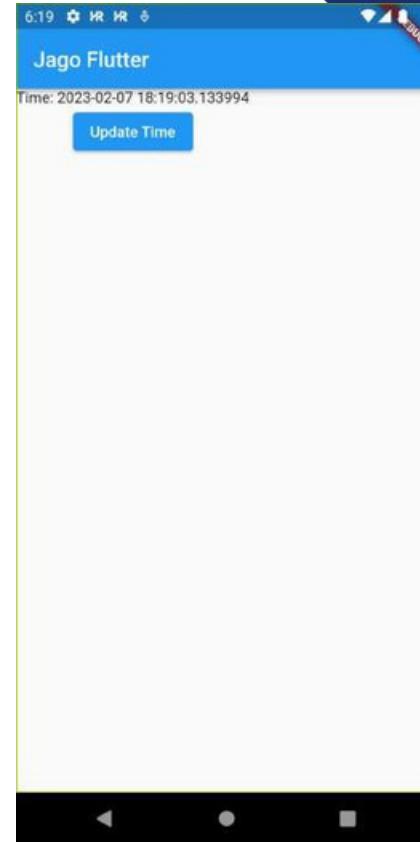
Class ini memiliki variable time dan valuenya dapat diganti ketika klik button pada line 34, setelah itu text akan di set ulang menggunakan DateTime.now() seperti baris 36, karena terdapat setState di situ, maka widget build akan di run ulang yang akhirnya menciptakan tampilan baru dengan data yang baru. Berikut hasilnya bila kita running dan klik button change text

```
1  class MyApp extends StatelessWidget {
2      const MyApp({super.key});
3
4      @override
5      Widget build(BuildContext context) {
6          return MaterialApp(
7              title: 'Flutter Demo',
8              theme: ThemeData(
9                  primarySwatch: Colors.blue,
10             ),
11             home: Scaffold(
12                 appBar: AppBar(
13                     title: const Text('Jago Flutter'),
14                 ),
15                 body: const ChangeTimeWidget(),
16             );
17         }
18     }
19
20 class ChangeTimeWidget extends StatefulWidget {
21     const ChangeTimeWidget({super.key});
22
23     @override
24     State<ChangeTimeWidget> createState() => _ChangeTimeWidgetState();
25 }
26
27 class _ChangeTimeWidgetState extends State<ChangeTimeWidget> {
28     DateTime time = DateTime.now();
29     @override
30     Widget build(BuildContext context) {
31         return Column(
32             children: [
33                 Text('Time: $time'),
34                 ElevatedButton(
35                     onPressed: () {
36                         time = DateTime.now();
37                         setState(() {});
38                     },
39                     child: const Text('Update Time'),
40                 ),
41             ],
42         );
43     }
44 }
```

## Stateful Widget

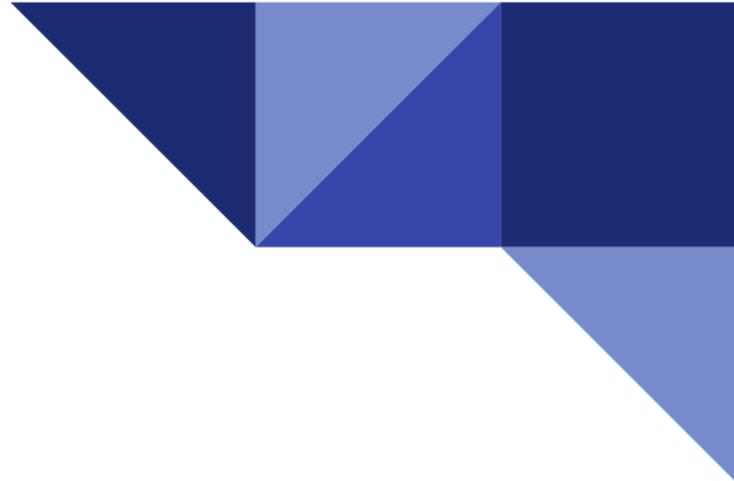


Click





ITB Stikom  
Ambon



# Terima Kasih