

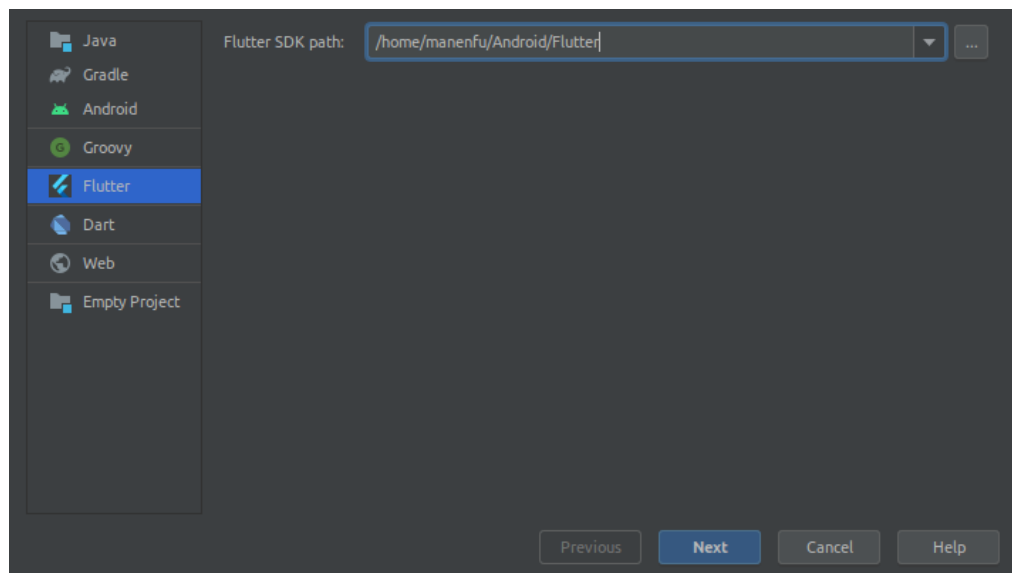
STRUKTUR PROJEK FLUTTER DAN APLIKASI *HELLO WORLD*

A. Materi

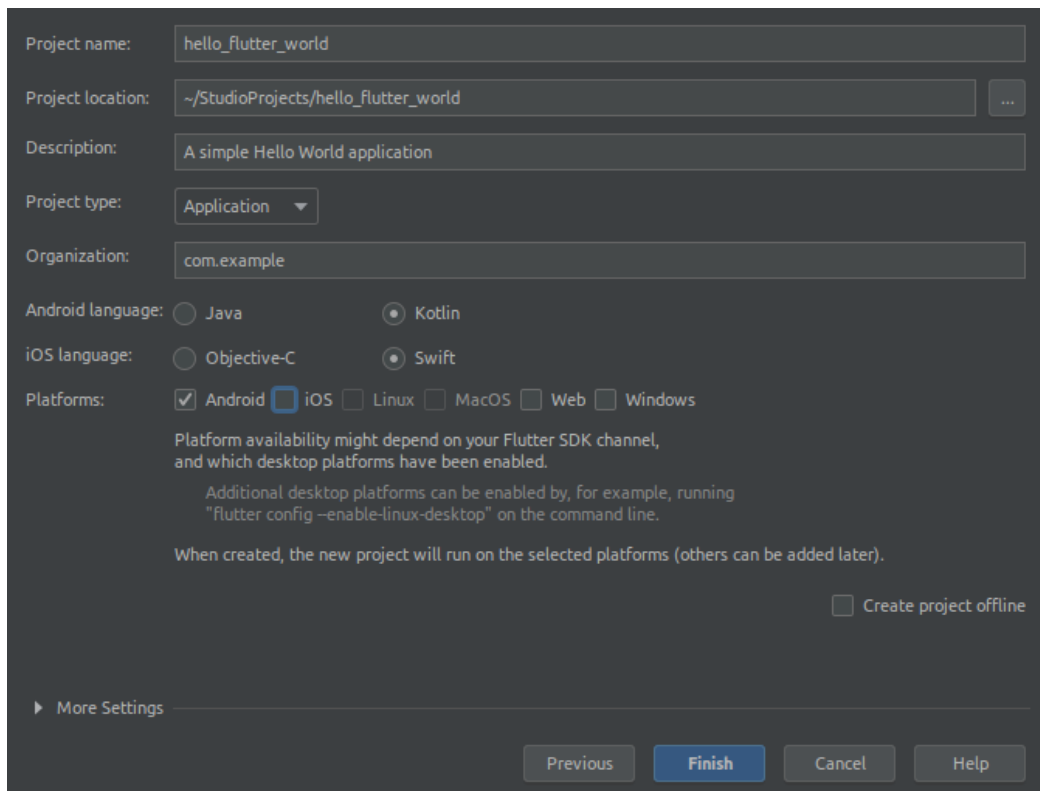
a. Membuat Proyek Flutter Baru

Flutter menyediakan *template* proyek yang dapat Anda gunakan untuk memulai proyek aplikasi Flutter.

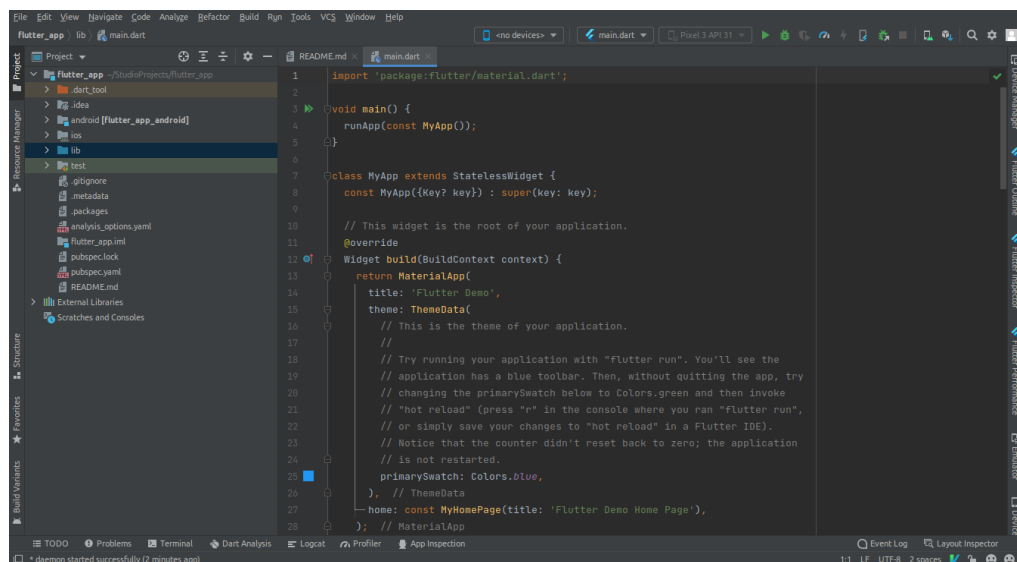
1. Pilih *File > New > New Flutter Project...*
2. Pilih Flutter, dan Sesuaikan *Flutter SDK Path* sesuai dengan lokasi folder `flutter` yang telah Anda pasang.



3. Tentukan nama proyek dan pilih *Application* sebagai jenis proyek. Kemudian, tekan *Finish*.



4. Tunggu beberapa saat. Android Studio akan membuat proyek baru. Anda dapat membaca kode program pada *file* lib/main.dart.



Pada modul ini Anda akan membuat aplikasi Flutter dari awal. Hapus semua konten pada *file* lib/main.dart sehingga menjadi *file* kosong.

b. Menulis Aplikasi *Hello World* dengan *Basic Widget*

User Interface pada aplikasi Flutter terdiri atas beberapa komponen yang disebut dengan *widget*. Setiap *widget* memiliki properti yang menentukan tampilan, letak, dan perilaku dari *widget* tersebut. *Widget* dapat mengandung *widget-widget* lain yang disebut dengan *children*. Pada Aplikasi *Hello World* yang akan dibuat, tiga jenis *widget* akan digunakan, yaitu `Text`, `Row/Column`, dan `Container`.

1. Text

Seperti namanya, *widget Text* akan menampilkan sebuah teks pada layar. Dalam membuat sebuah *widget Text*, Anda dapat mendeskripsikan teks yang akan ditampilkan, warna, dan *font* dari teks tersebut. Beberapa properti yang dimiliki *widget Text* antara lain:

- `data`: Menentukan isi dari teks
- `style`: Menentukan gaya dari teks. Hal ini mencakup warna, ukuran, dan ketebalan teks.
- `textDirection`: Menentukan arah penulisan teks. Untuk teks dengan huruf Latin, arah penulisan adalah dari kiri ke kanan (`ltr`).

2. Row dan Column

Widget ini digunakan untuk menampung dan menyusun *widget-widget* lain. *Row* akan menyusun *widget* di dalamnya secara horizontal dan *Column* akan menyusun *widget* secara vertikal. Beberapa properti yang dimiliki *widget Row* dan *Column* antara lain:

- `children`: Merupakan *list* dari *widget* anak yang akan disusun secara horizontal/vertikal
- `mainAxisAlignment`: Menentukan *alignment* dari penyusunan *children*.

3. Container

Widget ini digunakan untuk menampung *widget* lain. *Widget* ini berguna untuk melakukan manipulasi terhadap *child widget* secara bersamaan (seperti transformasi), atau membatasi ruang yang dapat diisi oleh *child widget*. Beberapa properti yang dimiliki *widget Container* antara lain:

- `color`: Menentukan warna latar belakang dari `Container`
- `child`: Merupakan *widget* anak yang dibungkus oleh `Container`

Untuk memulai pengembangan aplikasi Flutter, dibutuhkan suatu *boilerplate code*. Pada file `lib/main.dart`, tulis kode sebagai berikut:

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

void main() {
  runApp(
    // ...
  );
}
```

baris pertama akan mengimpor *library* yang dibutuhkan. `void main()` adalah fungsi utama yang akan dijalankan ketika aplikasi dijalankan. Fungsi ini akan memanggil fungsi `runApp()`. Fungsi `runApp()` menerima satu *widget* sebagai argumen dan akan menampilkan *widget* tersebut pada layar perangkat.

Pada aplikasi *Hello World* ini Anda akan mencoba membuat aplikasi yang menampilkan sebuah pesan '*Hello World*' dengan kata *Hello* dan *World* berada pada baris berbeda dan memiliki warna berbeda. Pertama-tama letakkan *widget* `Container` sebagai argumen dari fungsi `runApp()`. Penulisan *widget* seperti memanggil sebuah fungsi dengan daftar properti dari *widget* sebagai argumen.

```
void main() {
  runApp(
    Container(
      color: Colors.white,
      child: Column(
        // ...
      ),
    ),
  );
}
```

Kemudian buat *widget* `Column` sebagai child dari `Container`.

```

Container(
  color: Colors.white,
  child: Column(
    children: [
      // Tulis Text widget disini
    ],
    mainAxisAlignment: MainAxisAlignment.center,
  ),
)

```

Pengaturan nilai properti `mainAxisAlignment` agar *widget* anak dari `Column` disusun di tengah layar.

Kemudian buat dua buah *widget* `Text` sebagai elemen dari *list* `children` dari `Column`. `Text` pertama bertuliskan 'Hello,' dengan ukuran 40 dan warna hitam, sementara `Text` kedua bertuliskan 'World!' dengan ukuran 60 dan warna biru.

```

children: [
  Text(
    'Hello,',
    style: TextStyle(
      color: Colors.black,
      fontSize: 40.0,
    ),
    textDirection: TextDirection.ltr,
  ),
  Text(
    'World!',
    style: TextStyle(
      color: Colors.blue,
      fontWeight: FontWeight.bold,
      fontSize: 60.0,
    ),
    textDirection: TextDirection.ltr,
  ),
],

```

Kode Anda akan terlihat seperti ini.

```

import 'package:flutter/material.dart';

void main() {
  runApp(
    Container(
      color: Colors.white,
      child: Column(
        children: [
          Text(
            'Hello,',
            style: TextStyle(
              color: Colors.black,
              fontSize: 40.0,
            ),
            textDirection: TextDirection.ltr,
          ),
          Text(
            'World!',
            style: TextStyle(
              color: Colors.blue,
              fontWeight: FontWeight.bold,
              fontSize: 60.0,
            ),
            textDirection: TextDirection.ltr,
          ),
        ],
        mainAxisAlignment: MainAxisAlignment.center,
      ),
    ),
  );
}

```

Anda bisa mencoba menjalankan aplikasi ini dengan mengikuti petunjuk di bagian d. Aplikasi akan terlihat seperti berikut:

Hello,
World!

c. Material Component

Sebagian besar aplikasi Android menggunakan *Material design*. *Material design* merupakan bahasa dan gaya desain interaksi yang dikembangkan oleh Google untuk perangkat dengan layar sentuh. Untuk menggunakan *Material design* pada aplikasi *Hello World*, anda perlu membungkus *widget* yang telah Anda buat dalam beberapa komponen.:

1. MaterialApp

Widget ini membungkus *widget-widget* lain yang dibutuhkan dalam sebuah aplikasi *material design*. *Widget-widget* khusus untuk aplikasi *material design* harus menjadi keturunan dari sebuah *widget* `MaterialApp`. `MaterialApp` dapat menyimpan *widget-widget* ke beberapa “halaman” dan mengakses halaman-halaman tersebut menggunakan *routing*. Beberapa properti dari *widget* `MaterialApp` diantaranya:

- `title`: Judul aplikasi yang terlihat pada *task switcher* Android
- `home`: *widget* yang akan ditampilkan saat aplikasi dijalankan.

2. Scaffold

Widget ini memberikan *layout* dasar untuk memposisikan *widget-widget* yang umum ditemukan pada aplikasi *material design*, diantaranya:

- `appBar`: *Bar* di bagian atas layar

- `body`: Konten utama di tengah layar
- `bottomNavigationBar`: *Bar* di bagian bawah layar, biasanya untuk navigasi
- `floatingActionButton`: Tombol melayang yang berada di bagian bawah layar.

3. AppBar

Widget ini digunakan untuk membuat *App Bar*, yaitu *bar* yang berada di bagian atas layar. *AppBar* pada umumnya berisi nama aplikasi, nama halaman, tombol menu, dan tombol/*bar* pencarian.

Pada aplikasi ini `Container` yang berisi tulisan *Hello World* pada bagian b. dapat diletakkan sebagai *body* dari `Scaffold` dari sebuah `MaterialApp`. Pertama-tama, untuk memperbaiki organisasi dari kode sehingga lebih mudah dibaca, `Container` tersebut dapat didefinisikan di dalam sebuah kelas *widget* `HelloWidget`.


```

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

  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.white,
      constraints: BoxConstraints.expand(),
      child: Column(
        children: [
          Text(
            'Hello,',
            style: TextStyle(
              color: Colors.black,
              fontSize: 40.0,
            ),
          ),
          Text(
            'World!',
            style: TextStyle(
              color: Colors.blue,
              fontWeight: FontWeight.bold,
              fontSize: 60.0,
            ),
          ),
        ],
        mainAxisAlignment: MainAxisAlignment.center,
      ),
    );
  }
}

```

HelloWorldWidget merupakan subkelas dari StatelessWidget. Stateless Widget akan dijelaskan pada modul selanjutnya. Metode utama dari kelas HelloWorldWidget adalah metode `build()` yang akan membangun UI berdasarkan deskripsi *widget*. Pada kelas HelloWorldWidget, metode `build()` akan mengembalikan Container tersebut. Untuk menggunakan HelloWorldWidget, Anda cukup memanggil `HelloWorldWidget()`.

HelloWorldWidget akan disisipkan di dalam sebuah halaman yang didefinisikan di dalam kelas `HelloHomePage`.

```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        leading: IconButton(
          icon: Icon(Icons.menu),
          tooltip: 'Navigation',
          onPressed: null,
        ),
        title: Text('Hello Flutter World'),
        actions: [
          IconButton(
            icon: Icon(Icons.search),
            tooltip: 'Search',
            onPressed: null,
          ),
        ],
      ),
      body: HelloWidget(),
    );
  }
}

```

HelloHomePage membangun sebuah Scaffold yang mengandung AppBar berjudul 'Hello Flutter World' dengan tombol menu dan *search*, serta HelloWidget sebagai konten utama.

Ganti argumen fungsi runApp menjadi MaterialApp yang memiliki HelloHomePage sebagai *widget* utama.

```

void main() {
  runApp(
    MaterialApp(
      title: 'Hello Flutter World',
      home: HelloHomePage(),
    )
  );
}

```

d. Menjalankan Aplikasi

Anda dapat menjalankan aplikasi yang telah Anda buat di perangkat fisik Android atau di *emulator* perangkat Android.

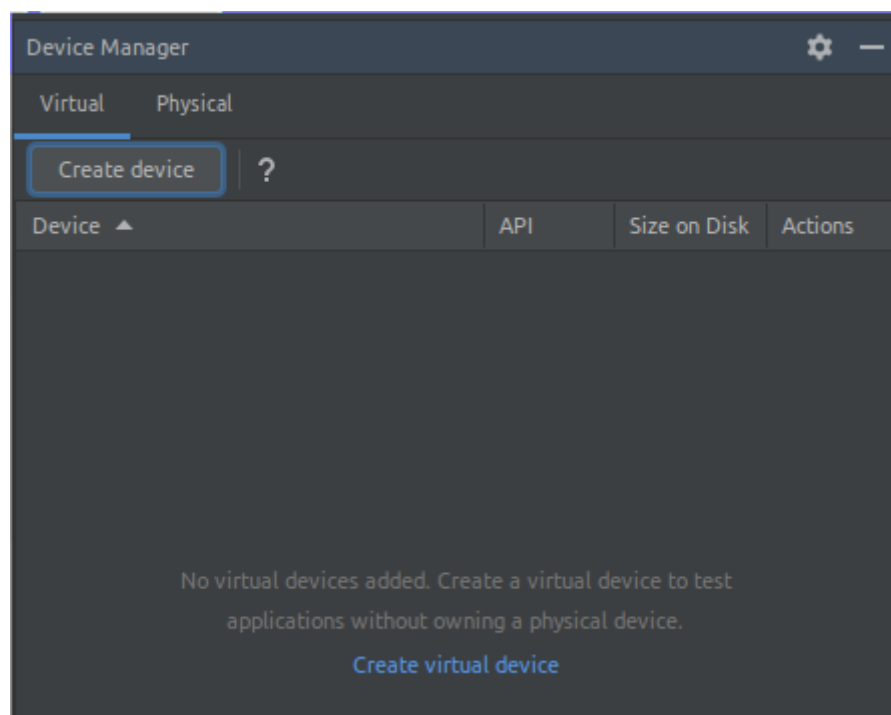
d.1. Menjalankan Aplikasi di Emulator

Jika Anda tidak memiliki perangkat Android, Anda dapat menggunakan *emulator* yang tersedia di Android Studio untuk menguji coba aplikasi. Selain itu, penggunaan *emulator* memungkinkan Anda menguji aplikasi di berbagai resolusi dan *form factor* perangkat. Perlu diperhatikan bahwa penggunaan *emulator* membutuhkan kemampuan komputer/laptop yang mumpuni.

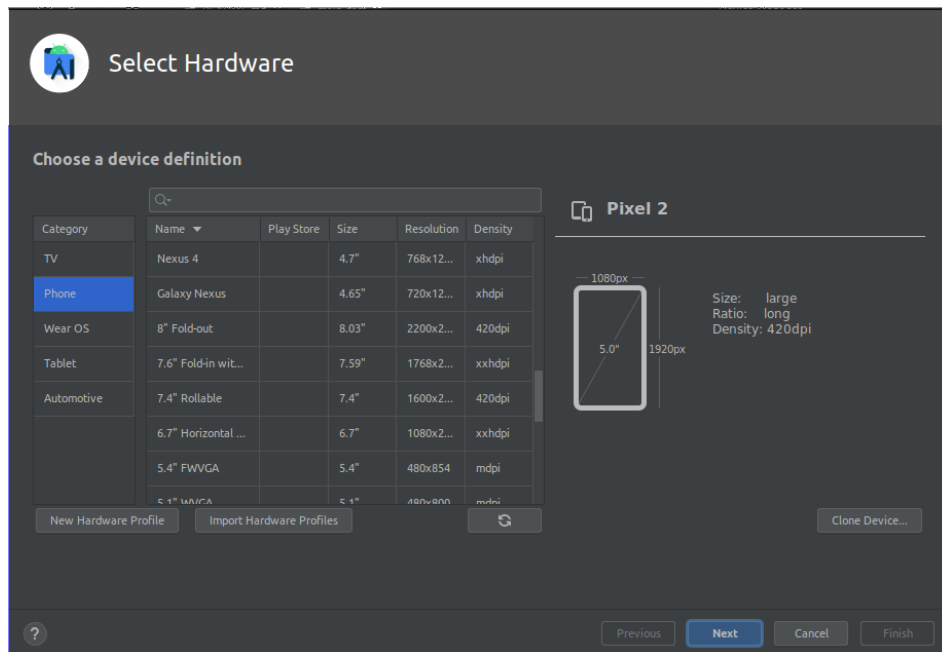
1. Buka *Device Manager*. Anda dapat menemukannya dengan menekan *icon* dengan logo Android pada toolbar, atau Anda dapat memilih menu *Tools > Device Manager* pada *menubar*.



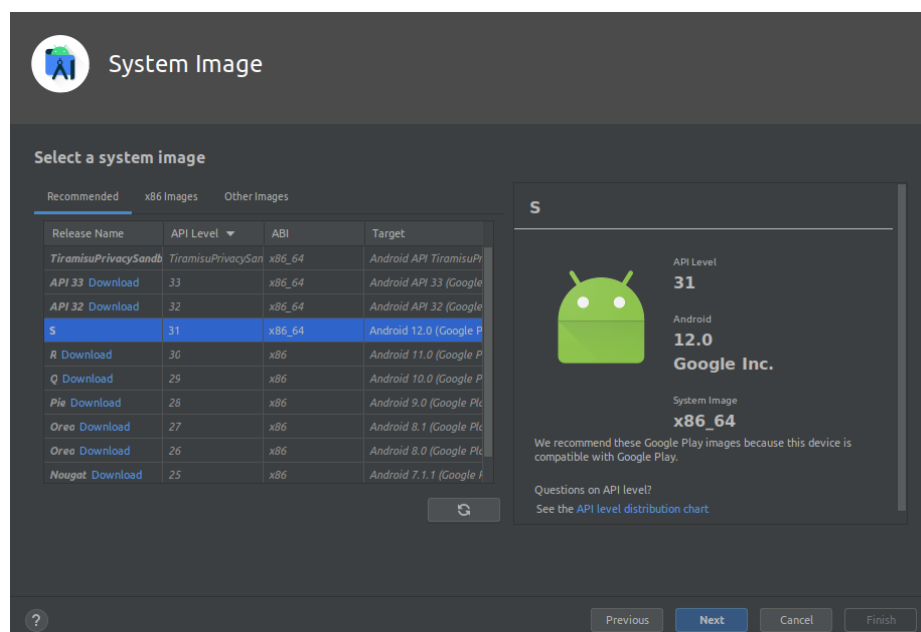
2. Pada panel *Device Manager*, tekan tombol *Create device*.



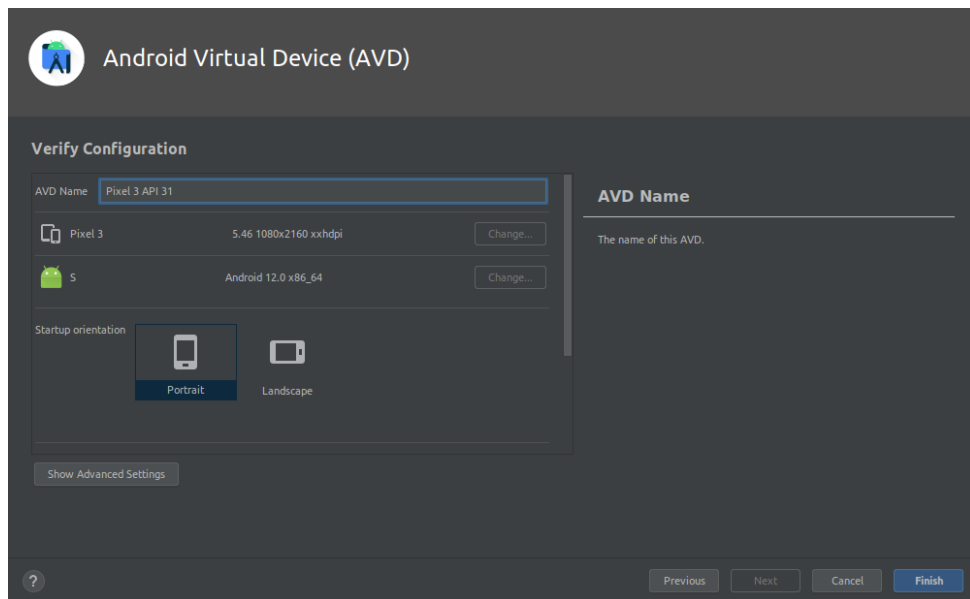
3. Pilih jenis perangkat yang Anda inginkan, lalu tekan *Next*.



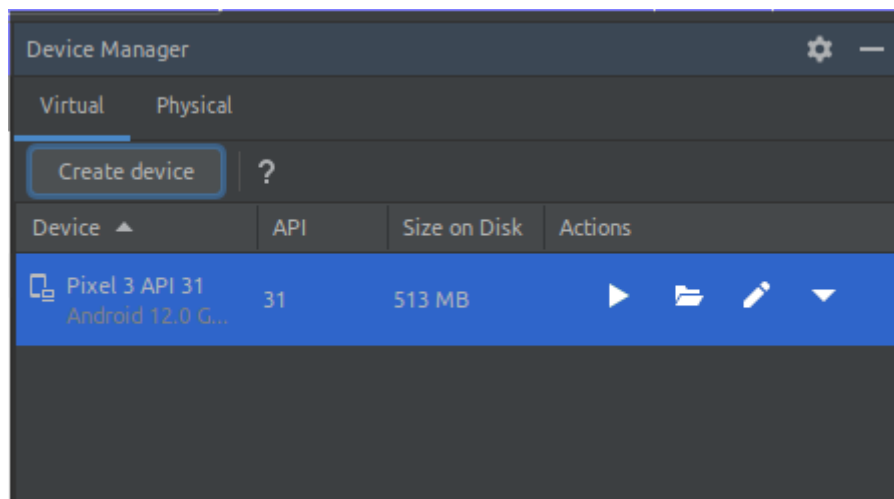
- Pilih versi Android yang akan dipasang di *emulator*. Anda perlu mengunduh versi Android tersebut sebelum menggunakannya dengan menekan *Download* di samping nama versi Android. Kemudian, tekan *Next*.



- Konfirmasi semua pilihan, lalu tekan *Finish*.



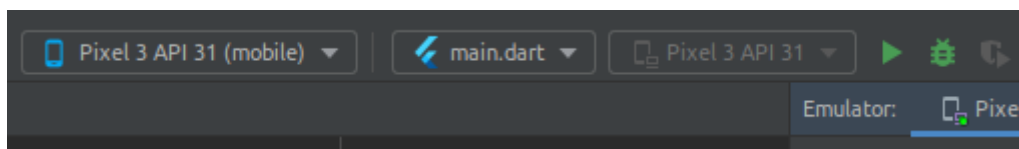
6. Perangkat baru akan terlihat di *Device Manager*.



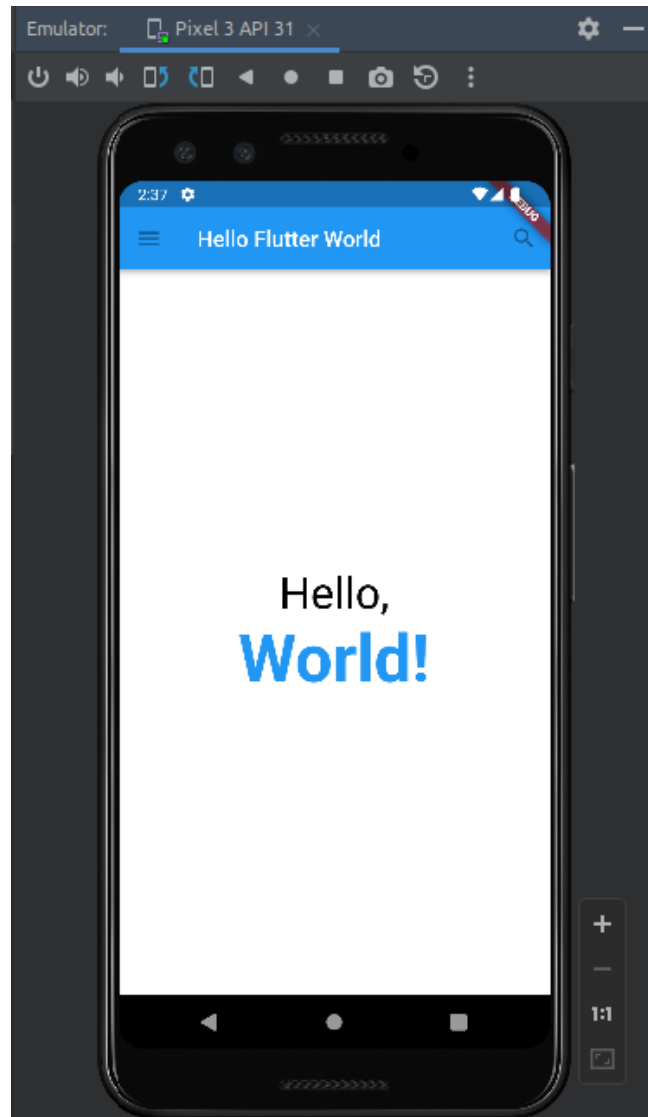
7. Tekan tombol *start* pada *Action* di samping *device*. *Emulator* akan dimulai.



8. Di bagian atas Android Studio, Anda dapat melihat *toolbar* yang memperlihatkan daftar perangkat yang terhubung.



9. Tekan tombol *play/run* untuk membangun, memasang, dan menjalankan aplikasi di perangkat Android Anda.
10. Setelah beberapa saat, aplikasi akan muncul dan berjalan di perangkat *emulator*.



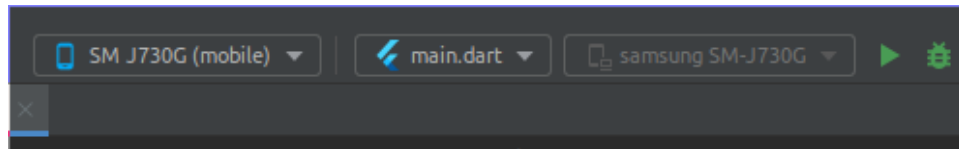
d.2. Menjalankan Aplikasi di perangkat Android

Untuk menjalankan sampel program pada *project* yang baru dibuat, Anda membutuhkan perangkat Android dan kabel USB.

1. Buka *project* Anda pada Android Studio
2. Jika Anda menggunakan Windows, Pastikan Anda membutuhkan *Google USB Driver*. *Driver* ini dapat diperoleh pada *Tools > SDK Manager*, lalu pilih *tab SDK Tools*, lalu centang *Google USB Driver* dan tekan *OK*.
3. Pada perangkat Android Anda, nyalakan *Developer Options* dan *USB Debugging*. Pada aplikasi *Settings* pada perangkat Android, pergi ke menu *About Phone* dan ketuk entri *Build Number* 7 kali. Setelah itu, menu *Developer Options* akan muncul di aplikasi *Settings*. Lokasi menu ini mungkin berbeda

pada tiap perangkat Android. Pada menu *Developer Options*, aktifkan opsi *USB Debugging*.

4. Hubungkan perangkat Android Anda ke komputer.
5. Di bagian atas Android Studio, Anda dapat melihat *toolbar* yang memperlihatkan daftar perangkat yang terhubung.



6. Tekan tombol *play/run* untuk membangun, memasang, dan menjalankan aplikasi di perangkat Android Anda.
7. Setelah beberapa saat, aplikasi akan muncul dan berjalan di perangkat Android Anda.



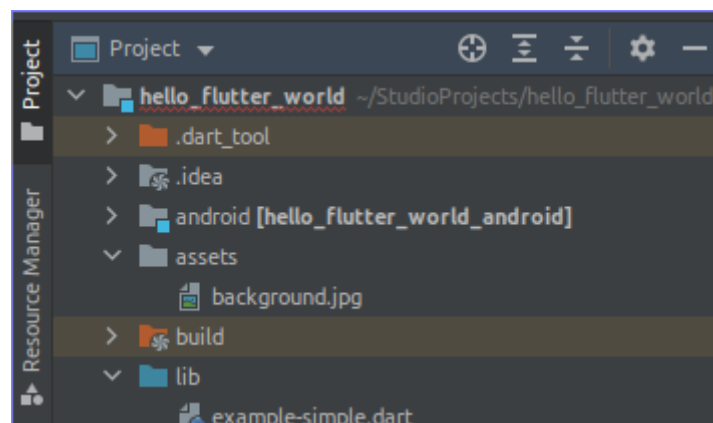
Hello,
World!

e. Menggunakan Asset pada Aplikasi *Hello World*

e.1. Menggunakan Asset Gambar pada Aplikasi *Hello World*

Saat ini, tampilan aplikasi masih sangat kosong. Anda dapat menambahkan elemen-elemen visual pada aplikasi, seperti gambar. Gambar adalah salah satu jenis *assets* yang dapat digunakan oleh aplikasi. Sebelumnya, Anda perlu menambahkan *asset* tersebut ke folder proyek Anda dan mendefinisikan *asset* tersebut dalam proyek Anda.

1. Dapatkan gambar secara bebas dari komputer atau internet (Pada modul ini bernama `background.jpg`). Buat folder `assets` di folder proyek Anda dan simpan gambar di folder `assets`.



2. Anda perlu membuat proyek Anda dapat mengenali folder `assets` tersebut. Hal ini dilakukan dengan mengubah file `pubspec.yaml`. File ini mendefinisikan beberapa hal penting pada proyek Anda, seperti dependensi (*library* tambahan yang digunakan) dan versi dari dependensi tersebut. Buka file `pubspec.yaml` dan di bawah entri `flutter:` buat entri `assets` dengan indentasi sebagai berikut:

```
# The following section is specific to Flutter.
flutter:

  # 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:
  # assets:
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg
  assets:
    - assets/
```

3. Sekarang Proyek dapat mengakses file dalam folder `assets`. Anda akan mencoba menggunakan gambar yang disimpan sebagai latar belakang pada tampilan aplikasi. Anda dapat mengakses `asset` menggunakan `Image.asset`.
4. Untuk membuat gambar sebagai latar belakang, Anda dapat meletakkan gambar di belakang `widget` `HelloWidget`. Hal ini dapat dicapai menggunakan `widget` `Stack`, yang menyusun `widget` anak secara menumpuk, dengan `widget` anak pertama berada paling belakang. Ganti properti `body` pada `HelloHomePage` menjadi `Stack` sebagai berikut:

```
class HelloHomePage extends StatelessWidget {
  HelloHomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // ...
      body: Stack(
        children: [
          Image.asset(
            'assets/background.jpg',
            fit: BoxFit.cover,
            height: double.infinity,
            width: double.infinity,
          ),
          HelloWidget(),
        ],
      ),
    );
  }
}
```

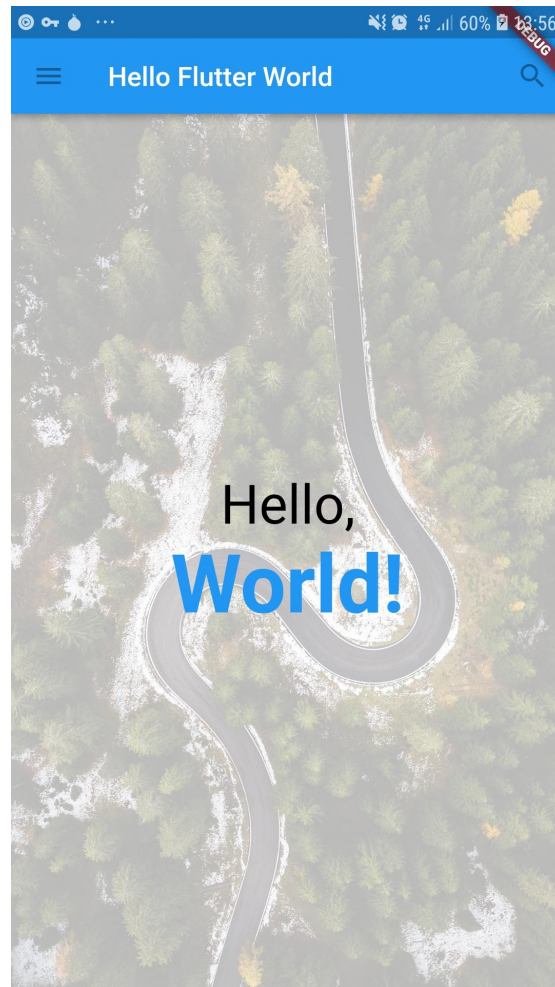
`Image.asset` menerima *path* dari *asset* gambar serta beberapa properti tambahan. Properti yang diberikan pada kode diatas agar gambar menutupi keseluruhan layar dan tidak terdistorsi.

5. Buat warna `Container` pada `HelloWidget` menjadi sedikit transparan agar gambar yang diletakkan di belakang terlihat.

```
class HelloWidget extends StatelessWidget {
  HelloWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.white70, // Translucent
      constraints: BoxConstraints.expand(),
      child: Column( ... ),
    );
  }
}
```









Tampilan layar akan menjadi sebagai berikut.



e.2. Menggunakan Asset Vektor pada Aplikasi *Hello World*

Vector image adalah gambar yang didefinisikan menggunakan bentuk-bentuk geometris pada ruang dua dimensi. Hal ini berbeda dengan *raster image* (gambar dengan format file .jpg, .png, dan sebagainya) yang terdiri dari kumpulan piksel. Format gambar vektor yang paling umum digunakan adalah format SVG (.svg). Salah satu keuntungan dari menggunakan gambar vektor adalah gambar tidak pecah ketika diubah ukurannya. Untuk dapat menggunakan *asset* gambar vektor pada proyek Anda, Anda membutuhkan *library* tambahan yaitu `flutter_svg`.

1. Pergi ke halaman https://pub.dev/packages/flutter_svg/versions dan pilih versi dari *library* yang akan digunakan. Perlu diperhatikan bahwa versi *library* harus kompatibel dengan versi Dart SDK. Versi Dart SDK yang digunakan sekarang dapat dilihat dengan menjalankan `flutter doctor --verbose` pada terminal atau memilih *Tools > Flutter > Flutter Doctor* pada Android Studio.

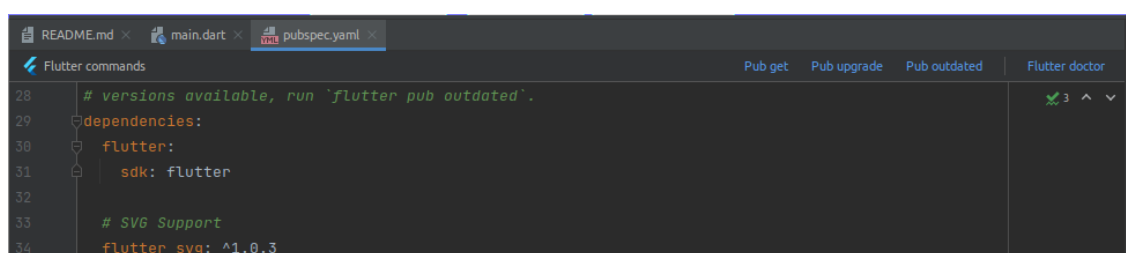
Stable versions of flutter_svg					
Version		Min Dart SDK	Uploaded		
1.1.1+1	Null safety	2.17	26 days ago		
1.1.0	Null safety	2.17(dev)	2 months ago		
1.0.3	Null safety	2.12	6 months ago		
1.0.2	Null safety	2.12	6 months ago		

2. Buka file `pubspec.yaml` dan tambahkan *library* `flutter_svg` dan versi yang dipilih pada entri `dependencies`.

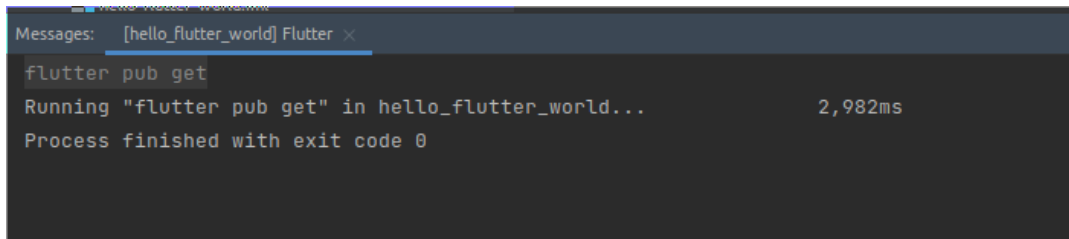
```
dependencies:
  flutter:
    sdk: flutter

  # SVG Support
  flutter_svg: ^1.0.3
```

3. Jalankan `flutter pub get` atau tekan tombol *Pub get* pada *editor* untuk mengunduh *library* tersebut.

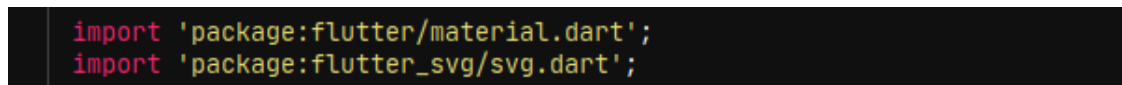


4. Jika proses berhasil, Anda akan menerima pesan berikut.



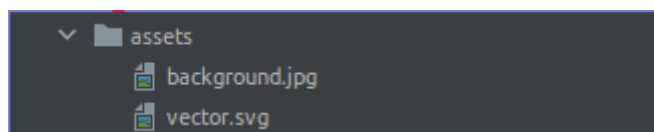
```
Messages: [hello_flutter_world] Flutter x
flutter pub get
Running "flutter pub get" in hello_flutter_world... 2,982ms
Process finished with exit code 0
```

5. Buka kembali file `lib/main.dart`. Di awal file, tambahkan baris untuk mengimpor *library* `flutter_svg`.



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

6. Siapkan gambar SVG yang akan digunakan dan simpan di folder `assets`. Pada modul ini gambar yang digunakan adalah `vector.svg`. Anda dapat mengakses *asset* SVG menggunakan `SvgPicture.asset()`



7. Pada file `lib/main.dart`, tambahkan *asset* SVG sebagai anak dari `Column` di `HelloWidget`. Kode ini akan menambahkan gambar SVG di atas teks 'Hello, World!'.

```

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

  @override
  Widget build(BuildContext context) {
    return Container(
      // ...
      child: Column(
        children: [
          SvgPicture.asset(
            'assets/vector.svg',
            height: 50.0,
            width: 50.0,
          ),
          Text(
            'Hello,',
            style: TextStyle(
              color: Colors.black,
              fontSize: 40.0,
            ),
          ),
          Text(
            'World!',
            style: TextStyle(
              color: Colors.blue,
              fontWeight: FontWeight.bold,
              fontSize: 60.0,
            ),
          ),
        ],
        mainAxisAlignment: MainAxisAlignment.center,
      ),
    );
  }
}

```

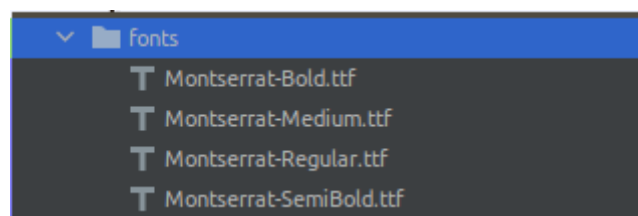
8. Jalankan aplikasi. Aplikasi akan terlihat sebagai berikut.



f. Menambah dan Mengubah *Font* pada Aplikasi *Hello World*

Anda dapat menggunakan *font* selain dari *font default* yang digunakan Android. Pertama-tama, Anda perlu menambahkan *font* tersebut pada proyek Anda.

1. Dapatkan *font* dengan format .ttf atau .otf dari komputer atau internet. Buat folder `fonts` pada folder proyek Anda dan simpan file *font* tersebut pada folder `fonts`.



2. Tambahkan definisi *font* di `pubspec.yaml` sebagai berikut.


```
flutter:
# ...

fonts:
- family: Montserrat
  fonts:
    - asset: fonts/Montserrat-Regular.ttf
      weight: 500
    - asset: fonts/Montserrat-Medium.ttf
      weight: 600
    - asset: fonts/Montserrat-SemiBold.ttf
      weight: 700
    - asset: fonts/Montserrat-Bold.ttf
      weight: 700
```

Pada umumnya sebuah *font* memiliki beberapa varian, seperti *weight* (Regular, Medium, Bold, dll). Varian ini diakses sebagai sebuah *font family*.

3. Untuk mengatur *font* secara global, Anda dapat menggunakan *theme* pada `MaterialApp`. tambahkan properti `theme` pada `MaterialApp` sebagai berikut.

```
runApp(
  MaterialApp(
    title: 'Hello Flutter World',
    theme: ThemeData(
      fontFamily: 'Montserrat',
    ),
    home: HelloHomePage(),
  )
);
```

4. Untuk mengatur *font* secara individual pada setiap teks, Anda dapat mengatur `fontFamily` pada properti `textStyle` pada *widget* `Text`.

```
Text(
  'Hello,',
  style: TextStyle(
    color: Colors.grey.shade800,
    fontSize: 40.0,
    fontFamily: 'Montserrat',
  ),
  textDirection: TextDirection.ltr,
),
```

5. Selanjutnya Anda dapat mengubah tampilan *widget* sesuai selera dengan mengatur properti *widget*. Kemudian jalankan aplikasi. Aplikasi Anda akan terlihat seperti berikut.



Jika Anda ingin mengeksplorasi *widget-widget* lain yang tersedia pada Flutter, Anda dapat mengakses dokumentasi Flutter di <https://docs.flutter.dev>.

B. Referensi

Flutter. 2022. *Flutter Documentation*. [Online] Internet: <https://docs.flutter.dev>. Diakses 27 Juli 2022.