

1.0: Pengantar Android

Materi:

- [Apa yang dimaksud dengan Android?](#)
- [Mengapa mengembangkan aplikasi untuk Android?](#)
- [Versi Android](#)
- [Tantangan development aplikasi Android](#)
- [Ketahui selengkapnya](#)

Apa yang dimaksud dengan Android?

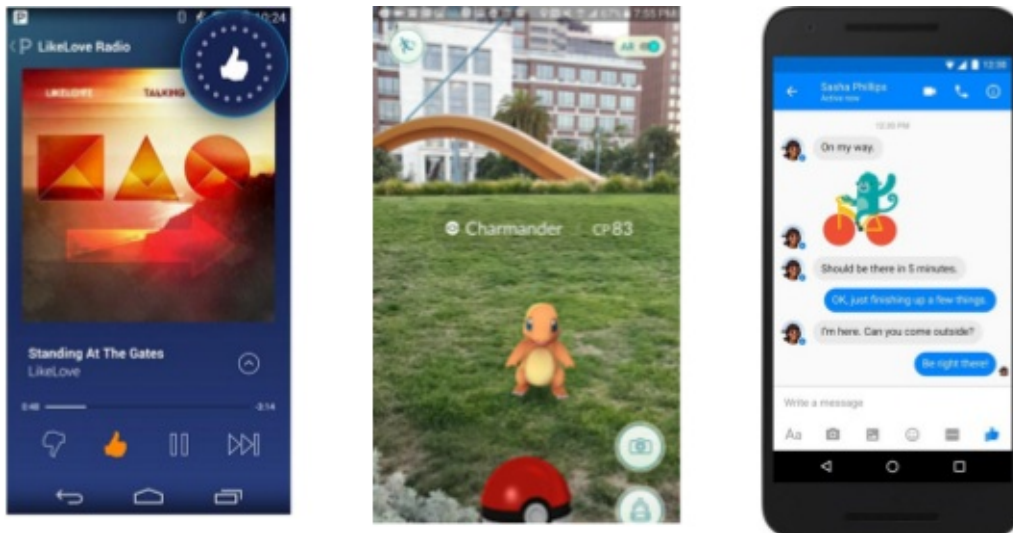
Android adalah sistem operasi dan platform pemrograman yang dikembangkan oleh Google untuk ponsel cerdas dan perangkat seluler lainnya (seperti tablet). Android bisa berjalan di beberapa macam perangkat dari banyak produsen yang berbeda. Android menyertakan kit development perangkat lunak untuk penulisan kode asli dan perakitan modul perangkat lunak untuk membuat aplikasi bagi pengguna Android. Android juga menyediakan pasar untuk mendistribusikan aplikasi. Secara keseluruhan, Android menyatakan ekosistem untuk aplikasi seluler.



Mengapa mengembangkan aplikasi untuk Android?

Aplikasi dikembangkan untuk berbagai alasan: menjawab kebutuhan bisnis, membangun layanan baru, membuat bisnis baru, dan menyediakan game serta jenis materi lainnya untuk pengguna. Developer memilih untuk mengembangkan bagi Android agar bisa menjangkau sebagian besar pengguna perangkat seluler.

Platform paling populer untuk aplikasi seluler



Pengalaman terbaik untuk pengguna aplikasi

Android menyediakan antarmuka pengguna (UI) layar sentuh untuk berinteraksi dengan aplikasi. Antarmuka pengguna Android sebagian besar berdasarkan pada manipulasi langsung, menggunakan isyarat sentuhan seperti menggesek, mengetuk, dan mencubit untuk memanipulasi objek di layar. Selain keyboard, ada keyboard virtual yang bisa disesuaikan untuk masukan teks. Android juga bisa mendukung pengontrol game dan keyboard fisik berukuran penuh yang dihubungkan dengan Bluetooth atau USB.



Layar utama Android bisa berisi sejumlah laman *ikon aplikasi*, yang akan meluncurkan aplikasi terkait, dan *widget*, dengan menampilkan materi langsung yang diperbarui secara otomatis seperti cuaca, kotak masuk email pengguna, atau ticker berita. Android juga bisa memutar materi multimedia seperti musik, animasi, dan video. Gambar di atas menampilkan ikon aplikasi pada layar utama (kiri), musik yang diputar (tengah), dan widget yang ditampilkan (kanan). Sepanjang bagian atas layar terdapat bilah status, yang menampilkan informasi tentang perangkat dan konektivitasnya. Layar utama Android bisa terdiri dari sejumlah laman, yang bisa digesek mundur dan maju oleh pengguna.

Android didesain untuk menyediakan respons cepat terhadap masukan pengguna. Selain antarmuka sentuh yang berubah-ubah, kemampuan getaran perangkat Android bisa menyediakan umpan balik sentuhan. Perangkat keras internal seperti akselerometer, giroskop, dan sensor kedekatan, digunakan oleh banyak aplikasi untuk merespons tindakan pengguna.

Android didesain untuk menyediakan respons cepat terhadap masukan pengguna. Selain antarmuka sentuh yang berubah-ubah, kemampuan getaran perangkat Android bisa menyediakan umpan balik sentuhan. Perangkat keras internal seperti akselerometer, giroskop, dan sensor kedekatan, digunakan oleh banyak aplikasi untuk merespons tindakan pengguna tambahan. Sensor tersebut bisa mendeteksi rotasi layar dari potret ke lanskap untuk tampilan yang lebih lebar atau sensor bisa memungkinkan pengguna untuk menyetir kendaraan virtual dengan memutar perangkat seolah-olah setir mobil.

Platform Android, berdasarkan pada [kernel Linux](#), terutama didesain untuk perangkat seluler layar sentuh seperti ponsel cerdas dan tablet. Karena perangkat Android biasanya bertenaga baterai, Android didesain untuk mengelola proses guna menjaga konsumsi daya tetap minimum, sehingga menyediakan penggunaan baterai lebih lama.

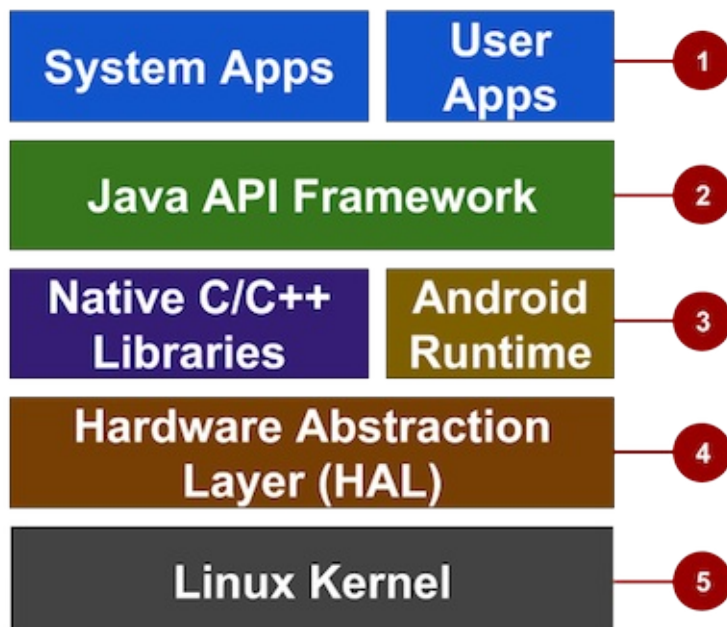
Mudah mengembangkan aplikasi

Gunakan Android Software Development Kit (SDK) Android untuk mengembangkan aplikasi yang memanfaatkan UI dan sistem operasi Android. SDK terdiri dari serangkaian alat development menyeluruh yang menyertakan debugger, pustaka perangkat lunak kode pratinjau, emulator perangkat, dokumentasi, kode contoh, dan tutorial. Gunakan alat-alat ini untuk membuat aplikasi yang terlihat hebat dan memanfaatkan kemampuan perangkat keras yang tersedia di setiap perangkat.

Untuk mengembangkan aplikasi menggunakan SDK, gunakan [bahasa pemrograman Java](#) untuk mengembangkan aplikasi dan file [Extensible Markup Language \(XML\)](#) untuk menjelaskan sumber daya data. Dengan menulis kode di Java dan membuat biner aplikasi tunggal, Anda akan memiliki aplikasi yang bisa berjalan pada faktor bentuk ponsel dan tablet. Anda bisa mendeklarasikan UI dalam rangkaian sumber daya XML ringan, satu rangkaian untuk bagian UI yang umum bagi semua faktor bentuk, dan rangkaian lain untuk fitur yang khusus bagi ponsel atau tablet. Pada waktu proses, Android menerapkan rangkaian sumber daya yang tepat berdasarkan ukuran layar, kepadatan, lokal, dan sebagainya.

Untuk membantu Anda mengembangkan aplikasi secara efisien, Google menawarkan Lingkungan Development Terintegrasi (IDE) Java lengkap yang disebut [Android Studio](#), dengan fitur lanjutan untuk pengembangan, debug, dan pemaketan aplikasi Android. Dengan menggunakan Android Studio, Anda bisa mengembangkan perangkat Android yang tersedia, atau membuat perangkat virtual yang mengemulasikan konfigurasi perangkat keras apa pun.

Android menyediakan arsitektur development yang kaya. Anda tidak perlu mengetahui banyak tentang komponen arsitektur ini, namun perlu mengetahui apa yang tersedia dalam sistem yang digunakan untuk aplikasi Anda. Diagram berikut menampilkan komponen utama sistem *tumpukan* Android — sistem operasi dan arsitektur development.



Dalam gambar di atas:

1. **Aplikasi:** Aplikasi berada pada tingkat ini, bersama dengan aplikasi sistem inti untuk email, perpesanan SMS, kalender, penjelajahan Internet, atau kontak.
2. **Kerangka Kerja API Java:** Semua fitur Android tersedia untuk developer melalui antarmuka pemrograman aplikasi

- [Sistem Tampilan](#) digunakan untuk membangun UI aplikasi, termasuk daftar, tombol, dan menu.
 - [Pengelola Referensi](#) digunakan untuk mengakses sumber daya non-kode seperti string, grafik, dan file layout yang dilokalkan.
 - [Pengelola Notifikasi](#) digunakan untuk menampilkan peringatan khusus di bilah status.
 - [Pengelola Aktivitas](#) yang mengelola daur hidup aplikasi.
 - [Penyedia Materi](#) yang memungkinkan aplikasi untuk mengakses data dari aplikasi lain.
 - Semua [API kerangka kerja](#) yang digunakan aplikasi sistem Android.
3. **Pustaka dan Waktu Proses Android:** Setiap aplikasi berjalan dalam prosesnya sendiri dan dengan instance Android Runtime sendiri, yang memungkinkan beberapa mesin sekaligus virtual pada perangkat bermemori rendah. Android juga menyertakan rangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8 yang digunakan kerangka kerja Java API. Banyak layanan dan komponen sistem Android inti dibangun dari kode asli yang memerlukan pustaka asli yang ditulis dalam C dan C++. Pustaka asli tersebut tersedia untuk aplikasi melalui kerangka kerja Java API.
4. **Hardware Abstraction Layer (HAL):** Lapisan ini menyediakan antarmuka standar yang menunjukkan kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth.
5. **Kernel Linux:** Fondasi platform Android adalah kernel Linux. Lapisan di atas mengandalkan kernel Linux untuk fungsionalitas pokok seperti threading dan manajemen memori tingkat rendah. Menggunakan kernel Linux memungkinkan Android memanfaatkan fitur keamanan utama dan memungkinkan produsen perangkat mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

Banyak opsi distribusi

Anda bisa mendistribusikan aplikasi Android dalam banyak cara: email, situs web, atau pasar aplikasi seperti [Google Play](#). Pengguna Android mengunduh jutaan aplikasi dan game dari [Google Play](#) store setiap bulan (ditampilkan dalam gambar di bawah ini). Google Play adalah layanan distribusi digital, yang dioperasikan dan dikembangkan oleh Google, yang berfungsi sebagai toko aplikasi resmi untuk Android, yang memungkinkan konsumen menjelajah dan mengunduh aplikasi


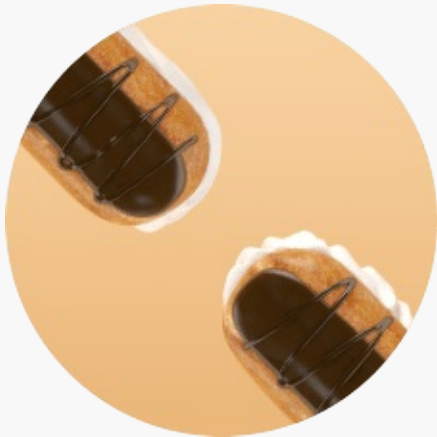


yang dikembangkan dengan Android SDK dan dipublikasikan melalui Google.




Versi Android

Versi Android

Google menyediakan peningkatan versi bertahap utama untuk sistem operasi Android setiap enam hingga sembilan bulan, menggunakan nama bertema makanan. Rilis utama yang terbaru adalah Android 7.0 "Nougat".

Nama kode	Nomor versi	Tanggal rilis awal	API level
N/A	1.0	23 September 2008	1
N/A	1.1	9 Februari 2009	2
Cupcake	1.5	27 April 2009	3
Donut 	1.6	15 September 2009	4
Eclair 	2.0 – 2.1	26 Oktober 2009	5–7
Froyo			

<p>Gingerbread</p>  A circular logo featuring a brown gingerbread man with a white smile and eyes, set against a pink circular background.	2.3 – 2.3.7	6 Desember 2010	9–10
<p>Honeycomb</p>  A circular logo with a yellow-to-orange gradient background. In the center is a honeycomb structure with a small pyramid of honeycomb cells on top.	3.0 – 3.2.6	22 Februari 2011	11–13
<p>Ice Cream Sandwich</p>  A circular logo with a light blue background. It shows a chocolate ice cream sandwich with white filling, resting on a silver stand with ice cubes.	4.0 – 4.0.4	18 Oktober 2011	14–15
<p>Jelly Bean</p>	4.1 – 4.3.1	9 Juli 2012	16–18

			
Jelly Bean			
	4.1 – 4.3.1	9 Juli 2012	16–18
KitKat			
	4.4 – 4.4.4	31 Oktober 2013	19–20
Lollipop			

	6.0 – 6.0.1	5 Oktober 2015	23
<p>Nougat</p> 	7.0	22 Agustus 2016	24

Lihat versi sebelumnya dan fiturnya dalam [Cerita Android](#).

[Dasbor untuk Versi Platform](#) diperbarui secara berkala untuk menampilkan distribusi perangkat aktif yang menjalankan setiap versi Android, berdasarkan jumlah perangkat yang mengunjungi Google Play Store. Mendukung sekitar 90% perangkat aktif adalah praktik yang baik, sekaligus menargetkan aplikasi Anda ke versi terbaru.

Catatan: Untuk menyediakan fitur dan fungsionalitas terbaik di seluruh versi Android, gunakan [Pustaka Dukungan Android](#) di aplikasi Anda. Pustaka dukungan ini memungkinkan aplikasi Anda menggunakan API platform terbaru di perangkat lama.

Tantangan development aplikasi Android

Sewaktu platform Android menyediakan fungsionalitas kaya untuk development aplikasi, masih ada sejumlah tantangan yang perlu Anda tangani, seperti:

- Membangun untuk dunia multilayar
- Mendapatkan kinerja yang tepat
- Membuat kode dan pengguna Anda tetap aman
- Tetap kompatibel dengan versi platform yang lebih lama
- Memahami pasar dan pengguna.

Membangun untuk dunia multilayar

Android berjalan pada miliaran perangkat genggam di seluruh dunia, dan mendukung beragam faktor bentuk termasuk perangkat yang dapat dikenakan dan televisi. Perangkat bisa tersedia dalam ukuran dan bentuk berbeda yang memengaruhi desain layar untuk elemen UI di aplikasi Anda.

- Mendapatkan kinerja yang tepat
- Membuat kode dan pengguna Anda tetap aman
- Tetap kompatibel dengan versi platform yang lebih lama
- Memahami pasar dan pengguna.

Membangun untuk dunia multilayar

Android berjalan pada miliaran perangkat genggam di seluruh dunia, dan mendukung beragam faktor bentuk termasuk perangkat yang dapat dikenakan dan televisi. Perangkat bisa tersedia dalam ukuran dan bentuk berbeda yang memengaruhi desain layar untuk elemen UI di aplikasi Anda.



Sebagai tambahan, produsen perangkat mungkin menambahkan elemen UI, gaya, dan warna sendiri untuk membedakan produk mereka. Setiap produsen menawarkan fitur berbeda dalam hal bentuk keyboard, ukuran layar, atau tombol kamera. Aplikasi yang berjalan pada satu perangkat mungkin terlihat sedikit berbeda di perangkat lain. Tantangan bagi sebagian besar developer adalah untuk mendesain elemen UI yang bisa bekerja di semua perangkat. Selain itu, developer juga bertanggung jawab untuk menyediakan sumber daya aplikasi seperti ikon, logo, grafik lain, dan gaya teks untuk mempertahankan keseragaman penampilan di seluruh perangkat yang berbeda.

Memaksimalkan kinerja aplikasi

Kinerja aplikasi, seberapa cepat aplikasi berjalan, seberapa mudah aplikasi menghubungkan ke jaringan, dan seberapa baik aplikasi mengelola baterai dan penggunaan memori, dipengaruhi oleh beberapa faktor seperti daya tahan baterai, materi multimedia, dan akses Internet. Anda harus memperhatikan batasan tersebut dan menulis kode sedemikian rupa sehingga penggunaan sumber daya diseimbangkan dan didistribusikan secara optimal. Misalnya, Anda harus menyeimbangkan layanan latar belakang dengan mengaktifkannya hanya jika perlu; hal ini akan menghemat daya tahan baterai perangkat pengguna.

Membuat kode dan pengguna Anda tetap aman

Anda perlu melakukan tindakan pencegahan untuk mengamankan kode dan pengalaman pengguna saat menggunakan aplikasi. Gunakan alat seperti ProGuard (disediakan di Android Studio), yang mendeteksi dan membuang kelas, bidang, metode, dan atribut yang tidak digunakan serta mengenkripsi semua kode dan sumber daya aplikasi sewaktu memaketkan aplikasi. Untuk melindungi informasi penting milik pengguna seperti proses masuk dan sandi, Anda harus mengamankan saluran komunikasi untuk melindungi data yang bergerak (di Internet) serta data yang tidak bergerak (di perangkat).

Tetap kompatibel dengan versi platform yang lebih lama

- [Ringkasan UI](#)
- [Versi Platform](#)
- [Mendukung Versi Platform Berbeda](#)
- Lainnya:
 - Panduan Pengguna Android Studio: [Image Asset Studio](#)
 - Wikipedia: [Rangkuman Riwayat Versi Android](#)

1.1: Buat Aplikasi Android Anda yang Pertama

Materi:

- Proses development
- Menggunakan Android Studio
- Menjelajahi proyek
- Menampilkan dan mengedit kode Java
- Menampilkan dan mengedit layout
- Memahami proses pembangunan
- Menjalankan aplikasi di emulator atau perangkat
- Menggunakan log
- Praktik terkait
- Ketahui selengkapnya

Bab ini menjelaskan cara mengembangkan aplikasi menggunakan Android Studio Integrated Development Environment (IDE).

Proses development

Proyek aplikasi Android dimulai dengan ide dan definisi persyaratan yang diperlukan untuk mewujudkan ide tersebut. Saat proyek berjalan, maka akan melalui desain, development, dan pengujian.

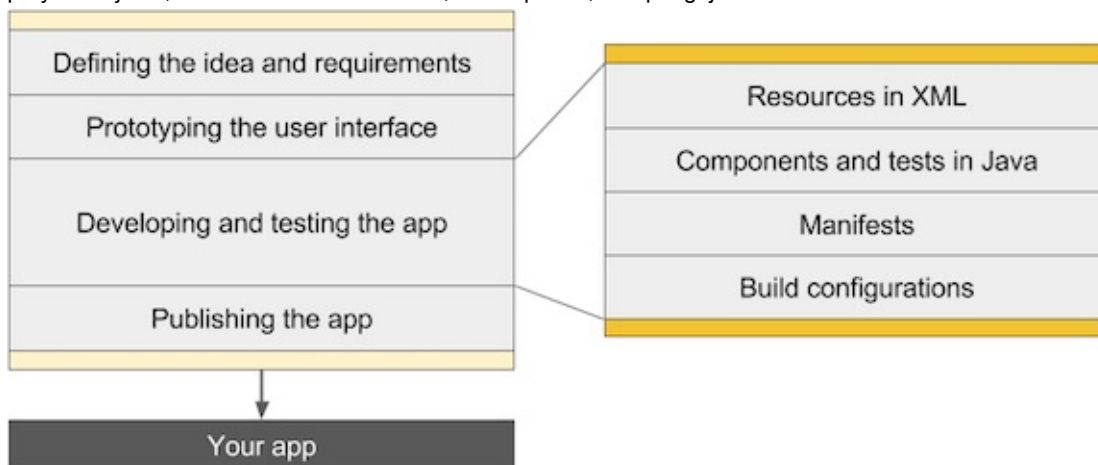


Diagram di atas adalah gambar proses development tingkat tinggi, dengan langkah-langkah berikut:

- *Mendefinisikan ide dan persyaratannya*: Sebagian besar aplikasi dimulai dengan ide tentang hal yang harus dilakukan, yang didukung pasar dan penelitian pengguna. Selama tahap ini, persyaratan aplikasi didefinisikan.
- *Membuat prototipe antarmuka pengguna*: Gunakan gambar, rekaan, dan prototipe untuk menampilkan tampilan antarmuka pengguna nantinya, dan cara kerjanya.
- *Mengembangkan dan menguji aplikasi*: Aplikasi terdiri dari satu atau beberapa aktivitas. Untuk setiap aktivitas, Anda bisa menggunakan Android Studio untuk melakukan hal berikut, tidak dalam urutan tertentu:
 - *Buat layout*: Tempatkan elemen UI pada layar di layout, dan tetapkan sumber daya string serta item menu, menggunakan Extensible Markup Language (XML).
 - *Tulis kode Java*: Buat referensi kode sumber untuk komponen dan pengujian, serta gunakan alat pengujian dan debug.
 - *Daftarkan aktivitas*: Deklarasikan aktivitas dalam file manifest.
 - *Definisikan versi*: Gunakan konfigurasi pembangunan default atau buat pembangunan khusus untuk versi aplikasi yang berbeda.
- *Mempublikasikan aplikasi*: Rakit APK (file paket) final dan distribusikan melalui saluran seperti Google Play.

Menggunakan Android Studio

Android Studio menyediakan alat untuk pengujian, dan mempublikasikan tahap proses development, serta lingkungan development terpadu untuk membuat aplikasi bagi semua perangkat Android. Lingkungan development menyertakan kode template dengan kode contoh untuk fitur aplikasi umum, alat pengujian dan kerangka kerja yang banyak, dan sistem pembangunan yang fleksibel.

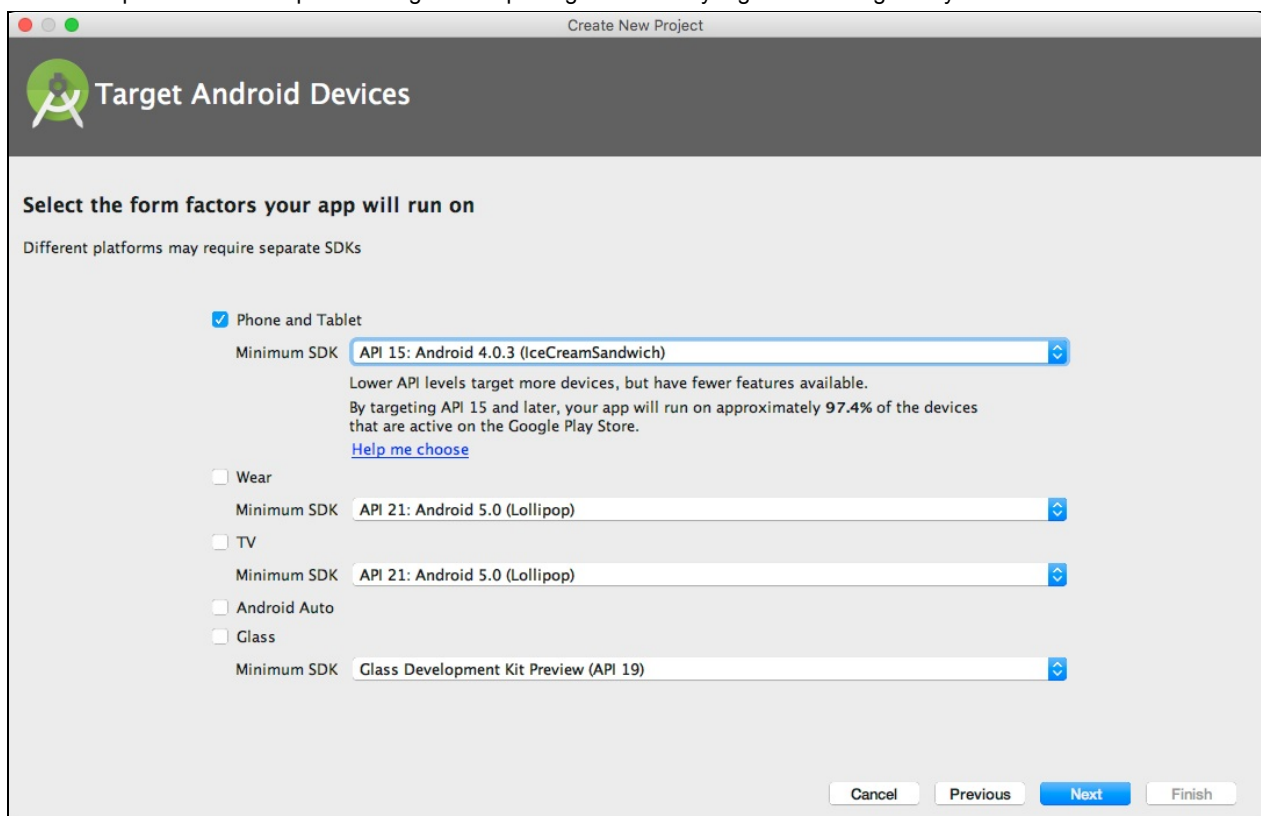
Memulai proyek Android Studio

Setelah berhasil memasang Android Studio IDE, klik dua kali ikon aplikasi Android Studio untuk memulainya. Pilih **Start a new Android Studio project** di jendela Welcome, dan beri nama proyek dengan nama yang sama yang ingin Anda gunakan untuk aplikasi.

Saat memilih Domain Perusahaan unik, ingat bahwa aplikasi yang dipublikasikan ke Google Play harus memiliki nama paket unik. Karena domain bersifat unik, yang mengawali nama aplikasi dengan nama Anda, atau nama domain perusahaan, sebaiknya sediakan nama paket unik yang memadai. Jika tidak berencana untuk mempublikasikan aplikasi, Anda bisa menerima domain contoh default. Ketahuilah bahwa mengubah nama paket di lain waktu memerlukan kerja tambahan.

Memilih perangkat target dan SDK minimum

Saat memilih Target Android Devices, Phone and Tablet dipilih secara default, seperti yang ditampilkan dalam gambar di bawah ini. Pilihan yang ditampilkan dalam gambar untuk Minimum SDK — **API 15: Android 4.0.3 (IceCreamSandwich)** — membuat aplikasi Anda kompatibel dengan 97% perangkat Android yang aktif di Google Play Store.



Perangkat berbeda menjalankan versi sistem Android yang berbeda, seperti Android 4.0.3 atau Android 4.4. Setiap versi yang berurutan umumnya menambahkan API baru yang tidak tersedia di versi sebelumnya. Untuk menunjukkan rangkaian API yang tersedia, setiap versi menetapkan API level. Misalnya, Android 1.0 adalah API level 1 dan Android 4.0.3 adalah API level 15.

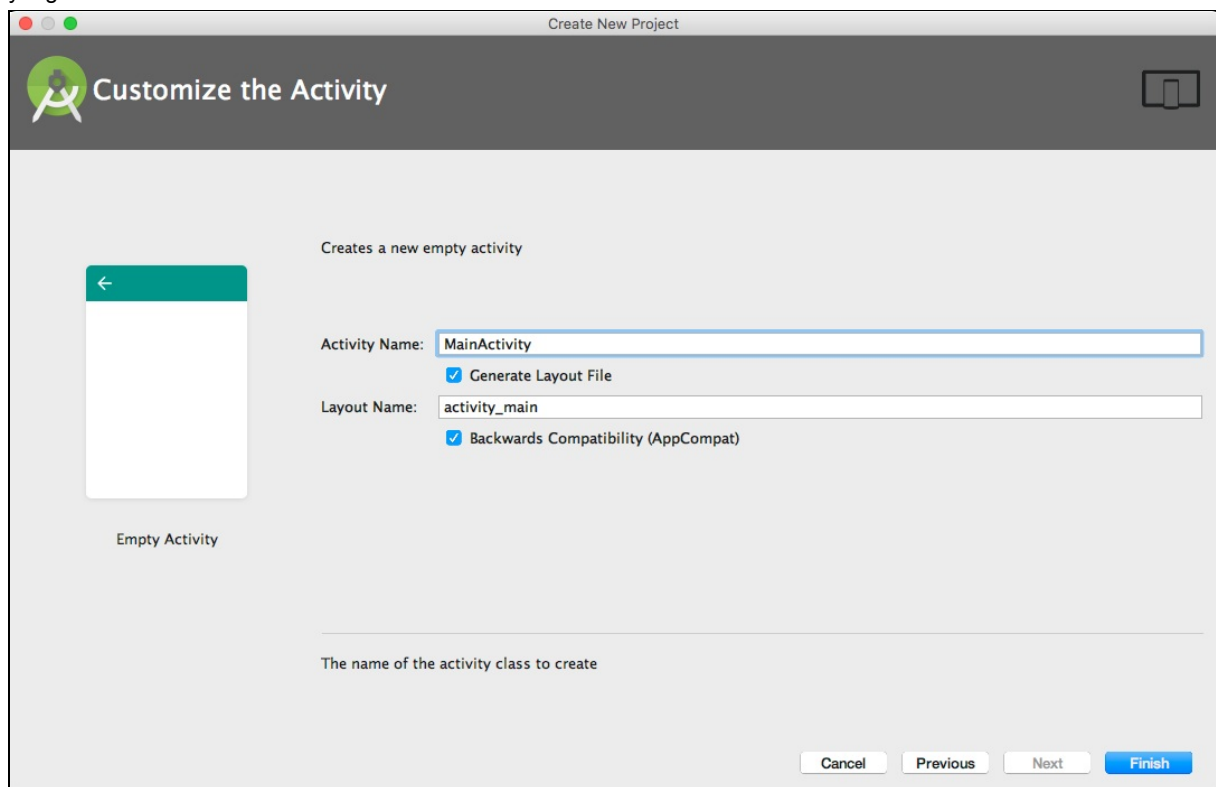
Minimum SDK mendeklarasikan versi Android minimum untuk aplikasi Anda. Setiap versi Android yang berurutan menyediakan kompatibilitas untuk aplikasi yang dibangun menggunakan API dari versi sebelumnya, sehingga aplikasi Anda akan *selalu* kompatibel dengan versi Android mendatang sambil menggunakan Android API yang didokumentasikan.

Memilih template

Android Studio mengumpulkan terlebih dulu proyek Anda dengan kode minimum untuk aktivitas dan layout layar berdasarkan *template*. Tersedia berbagai macam template, mulai dari template kosong virtual (Add No Activity) hingga beragam tipe aktivitas.

Anda bisa menyesuaikan aktivitas setelah memilih template. Misalnya, template Empty Activity menyediakan satu aktivitas yang disertai satu sumber daya layout untuk layar. Anda bisa memilih untuk menerima nama yang biasa digunakan untuk aktivitas (seperti **MainActivity**) atau mengubah nama di layar Customize Activity. Selain itu, jika Anda menggunakan template Empty Activity, pastikan memeriksa yang berikut ini jika belum diperiksa:

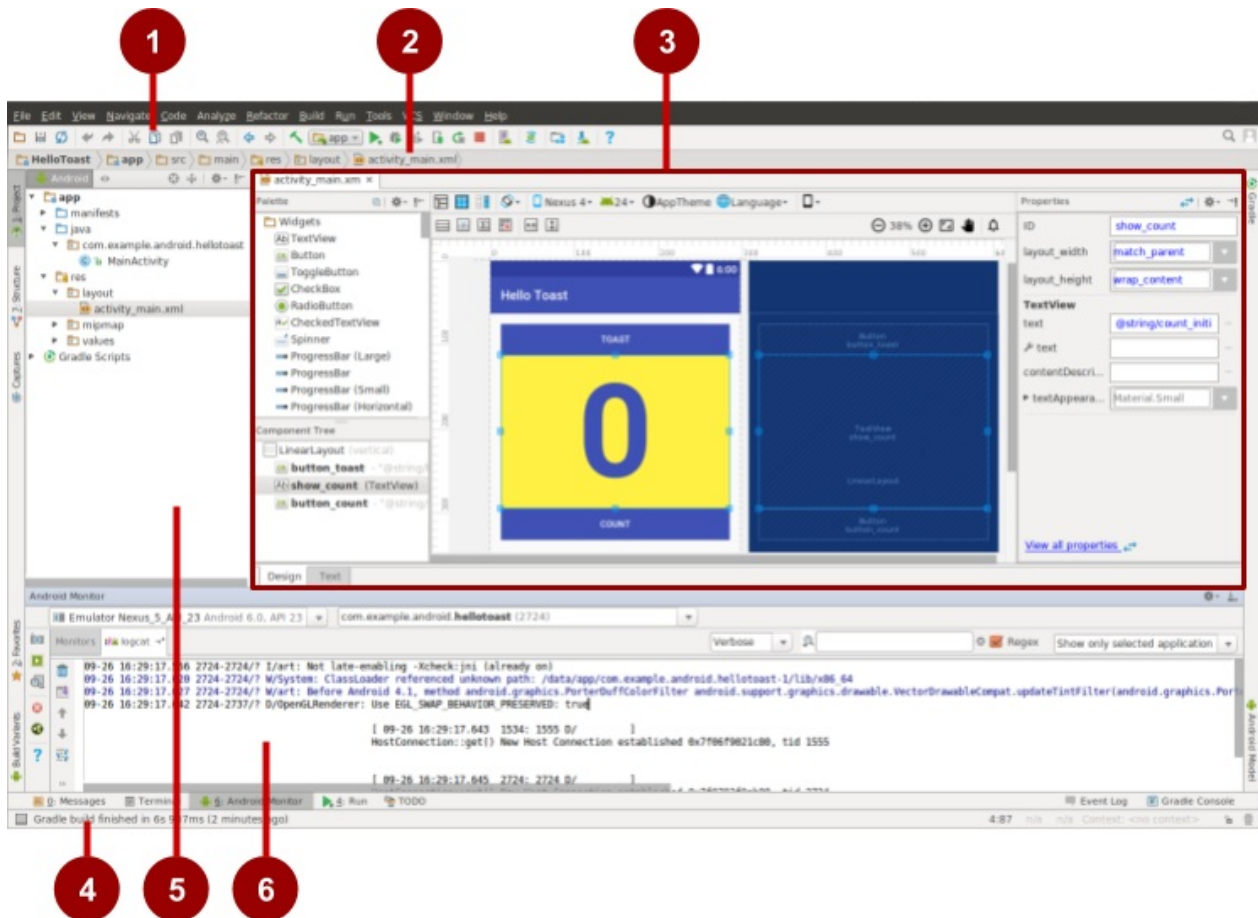
- **Generate Layout File:** Biarkan ini dicentang untuk membuat sumber daya layout yang terhubung dengan aktivitas ini, yang biasanya diberi nama **activity_main.xml**. Layout mendefinisikan antarmuka pengguna untuk aktivitas.
- **Backwards Compatibility (AppCompat):** Biarkan ini dicentang untuk menyertakan pustaka AppCompat sehingga aplikasi kompatibel dengan Android versi sebelumnya bahkan jika menggunakan fitur yang hanya ditemukan di versi yang lebih baru.



Android Studio membuat folder untuk proyek yang baru saja dibuat di folder AndroidStudioProjects pada komputer Anda.

Panel jendela Android Studio

Jendela utama Android Studio terdiri dari sejumlah area logis, atau *panel*, seperti yang ditampilkan dalam gambar di bawah ini.



Dalam gambar di atas:

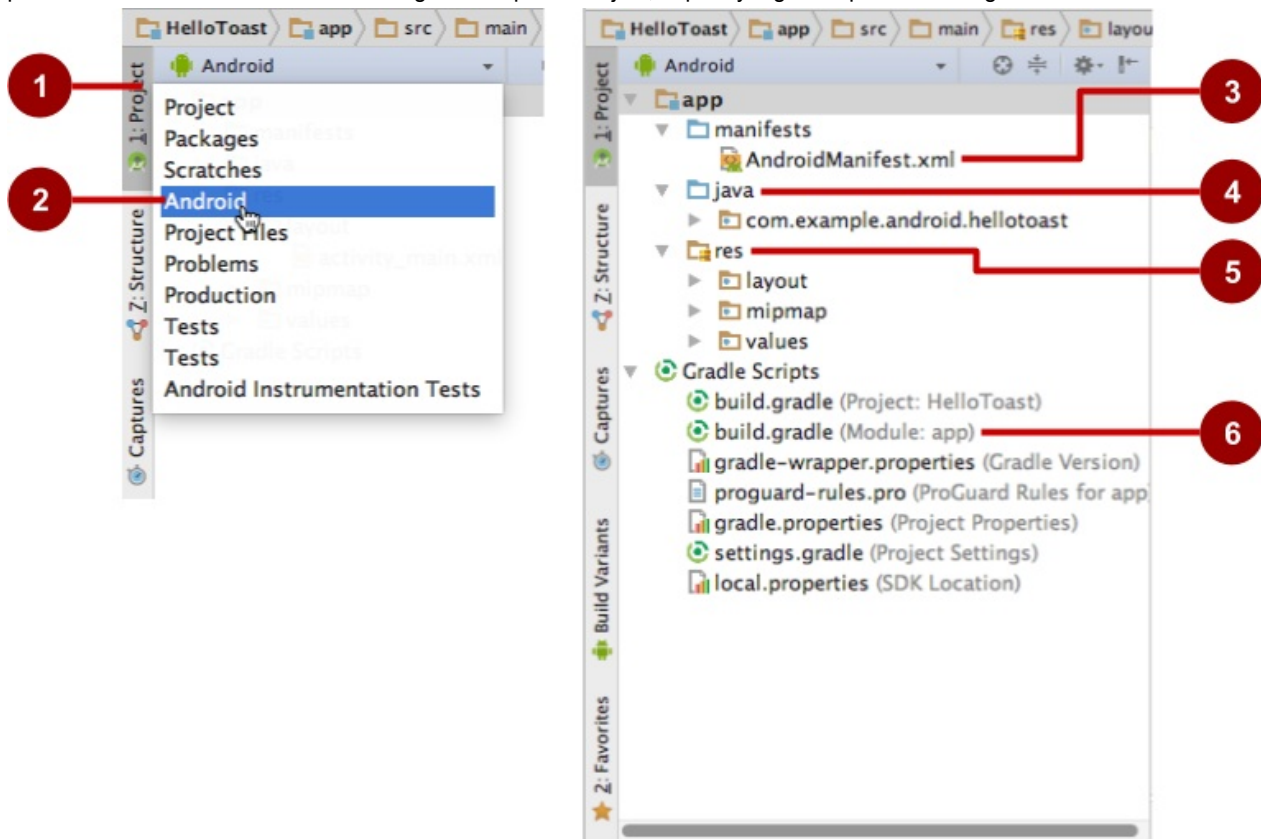
1. **Bilah Alat.** Bilah alat menjalankan beragam tindakan, termasuk menjalankan aplikasi Android dan meluncurkan alat Android.
2. **Bilah Navigasi.** Bilah navigasi memungkinkan navigasi melalui proyek dan membuka file untuk pengeditan. Bilah navigasi menyediakan tampilan struktur proyek yang lebih ringkas.
3. **Panel Editor.** Panel ini menampilkan materi file yang dipilih dalam proyek. Misalnya, setelah memilih layout (seperti yang ditampilkan dalam gambar), panel ini menampilkan editor layout dengan alat untuk mengedit layout. Setelah memilih file kode Java, panel ini menampilkan kode dengan alat untuk mengedit kode.
4. **Bilah Status.** Bilah status menampilkan status proyek dan Android Studio itu sendiri, serta peringatan atau pesan apa pun. Anda bisa mengamati kemajuan pembangunan di bilah status.
5. **Panel Project.** Panel proyek menampilkan file proyek dan hierarki proyek.
6. **Panel Monitor.** Panel monitor menawarkan akses ke daftar TODO untuk mengelola tugas, Android Monitor untuk memantau eksekusi aplikasi (ditampilkan dalam gambar), logcat untuk menampilkan pesan log, dan aplikasi Terminal untuk melakukan aktivitas Terminal.

Tip: Anda bisa mengatur jendela utama untuk memberi Anda lebih banyak ruang layar dengan menyembunyikan atau memindahkan panel. Anda juga bisa menggunakan pintasan keyboard untuk mengakses lebih banyak fitur. Lihat [Pintasan Keyboard](#) untuk daftar lengkap.

Menjelajahi proyek

Setiap proyek di Android Studio berisi file `AndroidManifest.xml`, file kode sumber komponen, dan file sumber daya terkait. Secara default, Android Studio mengatur file proyek Anda berdasarkan jenis file, dan menampilkannya dalam proyek: Tampilan Android di panel alat (bantu) kiri, seperti yang ditampilkan di bawah ini. Tampilan menyediakan akses cepat ke file kunci proyek Anda.

Untuk beralih kembali ke tampilan ini dari tampilan lainnya, klik tab **Project** vertikal di kolom paling kiri panel Project, dan pilih **Android** dari menu munculan di bagian atas panel Project, seperti yang ditampilkan dalam gambar di bawah ini.



Dalam gambar di atas:

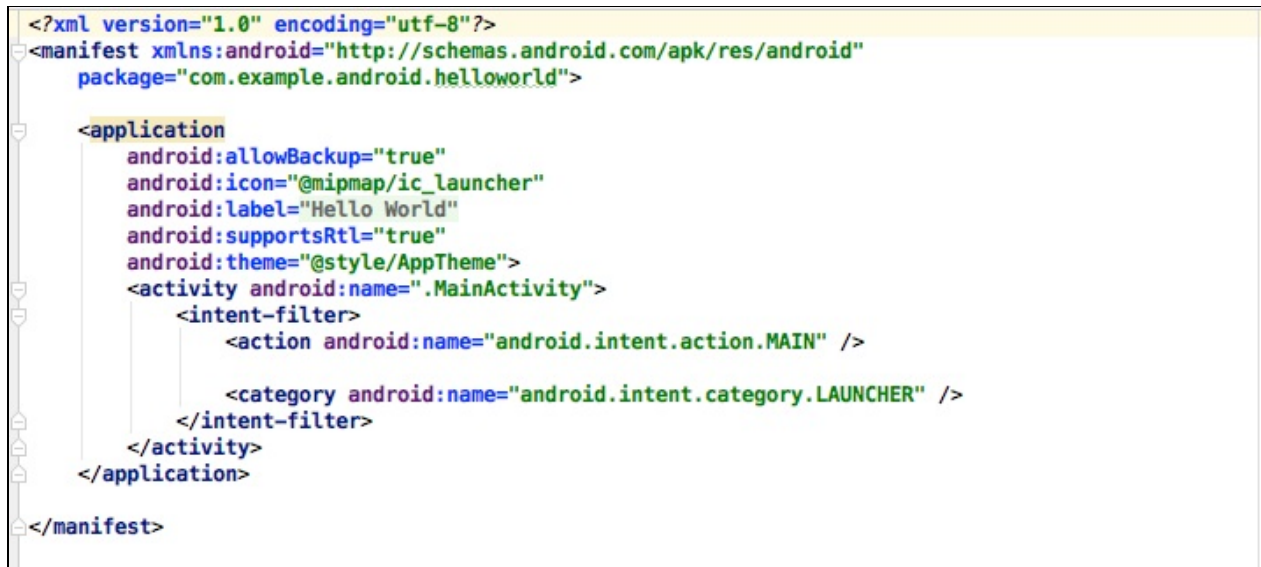
1. Tab **Project** . Klik untuk menampilkan tampilan proyek.
2. Pilihan **Android** di menu tarik-turun proyek.
3. File **AndroidManifest.xml**. Digunakan untuk menetapkan informasi tentang aplikasi bagi lingkungan waktu proses Android. Template yang Anda pilih membuat file ini.
4. Folder **java**. Folder ini menyertakan aktivitas, pengujian, dan komponen lain di kode sumber Java. Setiap aktivitas, layanan, dan komponen lain didefinisikan sebagai kelas Java, biasanya dalam file miliknya. Nama aktivitas pertama (layer) yang pengguna lihat, yang juga melakukan inisialisasi sumber daya seluruh aplikasi, biasanya adalah MainActivity.
5. Folder **res**. Folder ini menyimpan sumber daya, seperti layout XML, string UI, dan gambar. Aktivitas biasanya dikaitkan dengan file sumber daya XML yang menetapkan layout tampilannya. File ini biasanya diberi nama setelah aktivitas atau fungsinya.
6. File **build.gradle (Module: App)**
7. **. File ini menetapkan konfigurasi pembangunan modul. Template yang Anda pilih membuat file ini, yang mendefinisikan konfigurasi pembangunan, termasuk atribut `minSdkVersion` yang mendeklarasikan versi minimum untuk aplikasi, dan atribut `targetSdkVersion` yang mendeklarasikan versi tertinggi (terbaru) untuk aplikasi yang dioptimalkan. File ini juga menyertakan daftar dependensi*, yaitu pustaka yang diperlukan oleh kode — seperti pustaka AppCompatActivity untuk mendukung beragam versi Android.*

1

Menampilkan Manifes Android

Sebelum sistem Android bisa memulai komponen aplikasi, sistem harus mengetahui bahwa komponen ada dengan membaca file **AndroidManifest.xml** aplikasi. Aplikasi harus mendeklarasikan semua komponennya dalam file ini, yang harus berada pada akar direktori proyek aplikasi.

Untuk menampilkan file ini, luaskan folder manifest di Project: Tampilan Android, dan klik dua kali file (**AndroidManifest.xml**). Isinya muncul di panel pengeditan seperti yang ditampilkan dalam gambar di bawah ini.



Namespace dan tag aplikasi Android

Manifest Android dikodekan di XML dan selalu menggunakan namespace Android:

```
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.android.helloworld">
```

Eksresi `package` menampilkan nama paket unik aplikasi baru. Jangan mengubah ini setelah aplikasi dipublikasikan.

```
<application
...
</application>
```

Tag `<application`, dengan tag `</application>` penutupnya, mendefinisikan setelan manifest untuk keseluruhan aplikasi.

Cadangan otomatis

Atribut `android:allowBackup` memungkinkan pencadangan data aplikasi otomatis:

```
...
android:allowBackup="true"
...
```

Menyetel atribut `android:allowBackup` ke `true` memungkinkan aplikasi dicadangkan secara otomatis dan dipulihkan jika perlu. Pengguna menggunakan waktu dan upaya untuk mengonfigurasi aplikasi. Beralih ke perangkat baru bisa membatalkan seluruh konfigurasi yang dibuat dengan susah payah. Sistem melakukan cadangan otomatis ini untuk hampir seluruh data aplikasi secara default, dan melakukannya tanpa mengharuskan developer menulis kode aplikasi tambahan lainnya.

Untuk aplikasi dengan versi SDK target berupa Android 6.0 (API level 23) dan yang lebih tinggi, perangkat yang menjalankan Android 6.0 dan yang lebih tinggi secara otomatis membuat cadangan data aplikasi ke awan karena atribut `android:allowBackup` default ke `true` jika dihilangkan. Untuk aplikasi < API level 22, Anda harus secara eksplisit menambahkan atribut `android:allowBackup` dan menyetelnya ke `true`.

Tip: Untuk mengetahui selengkapnya tentang pencadangan otomatis untuk aplikasi, lihat [Mengonfigurasi Cadangan Otomatis untuk Aplikasi](#).

Ikon aplikasi

Metode `android:icon` menyetel ikon untuk aplikasi:

```
...
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
...
```

Atribut `android:icon` menetapkan ikon di folder **mipmap** (di dalam folder **res** di Project: tampilan Android) untuk aplikasi. Ikon muncul di Peluncur untuk meluncurkan aplikasi. Ikon juga digunakan sebagai ikon default untuk komponen aplikasi.

Sumber daya string dan label aplikasi

Seperti yang bisa Anda lihat di gambar sebelumnya, atribut `android:label` menampilkan string `"Hello World"` yang disorot. Jika Anda mengeklik string ini, maka string berubah untuk menampilkan sumber daya string `@string/app_name` :

```
...
android:label="@string/app_name"
...
```

Tip: Ctrl - klik atau klik-kanan `app_name` di panel edit untuk melihat menu konteks. Pilih **Go To > Declaration** untuk melihat lokasi sumber daya string dideklarasikan: dalam file `strings.xml`. Bila Anda memilih **Go To > Declaration** atau membuka file dengan mengeklik dua kali **strings.xml** di Project: Tampilan Android (di dalam folder **values**), isinya muncul di panel pengeditan.

Setelah membuka file `strings.xml`, Anda bisa melihat bahwa nama string `app_name` disetel ke `Hello World` . Anda bisa mengubah nama aplikasi dengan mengubah string `Hello World` ke hal lain. Sumber daya string dijelaskan dalam pelajaran terpisah.

Tema aplikasi

Atribut `android:theme` menyetel tema aplikasi, yang mendefinisikan penampilan elemen antarmuka pengguna seperti teks:

```
...
android:theme="@style/AppTheme">
...
```

Atribut `theme` disetel ke tema standar `AppTheme` . Tema dijelaskan dalam pelajaran terpisah.

Mendeklarasikan versi Android

Perangkat yang berbeda mungkin menjalankan versi sistem Android yang berbeda, seperti Android 4.0 atau Android 4.4. Setiap versi yang berurutan bisa menambahkan API baru yang tidak tersedia di versi sebelumnya. Untuk menunjukkan rangkaian API yang tersedia, setiap versi menetapkan API level. Misalnya, Android 1.0 adalah API level 1 dan Android 4.4 adalah API level 19.

API level memungkinkan developer untuk mendeklarasikan versi minimum dengan aplikasi yang kompatibel, menggunakan tag manifest `<uses-sdk>` dan atribut `minSdkVersion` . Misalnya, Calendar Provider API telah ditambahkan di Android 4.0 (API level 14). Jika aplikasi Anda tidak berfungsi tanpa API tersebut, deklarasikan API level 14 sebagai versi yang didukung minimum aplikasi seperti ini:

```
<manifest ... >
    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="19" />
    ...
</manifest>
```

Atribut `minSdkVersion` mendeklarasikan versi minimum untuk aplikasi, dan atribut `targetSdkVersion` mendeklarasikan versi tertinggi (terbaru) yang telah dioptimalkan dalam aplikasi. Setiap versi Android yang berurutan menyediakan kompatibilitas untuk aplikasi yang dibangun menggunakan API dari versi sebelumnya, sehingga aplikasi akan *selalu* kompatibel dengan versi Android mendatang sambil menggunakan Android API yang didokumentasikan.

Atribut `targetSdkVersion` *tidak* mencegah aplikasi dipasang pada versi Android yang lebih tinggi (yang lebih baru) dibandingkan nilai yang ditetapkan, namun hal ini penting karena menunjukkan pada sistem apakah aplikasi harus mewarisi perubahan perilaku di versi yang baru. Jika Anda tidak memperbarui `targetSdkVersion` ke versi terbaru, sistem akan menganggap bahwa aplikasi memerlukan perilaku kompatibilitas mundur saat dijalankan pada versi terbaru. Misalnya, di antara perubahan perilaku di Android 4.4, alarm yang dibuat dengan `AlarmManager` API saat ini tidak tepat secara default sehingga sistem bisa mengumpulkan alarm aplikasi dan mempertahankan kekuatan sistem, namun sistem akan menahan perilaku API sebelumnya untuk aplikasi jika API level target Anda lebih rendah daripada "19".

Menampilkan dan mengedit kode Java

Komponen ditulis di Java dan dicantumkan dalam folder modul di folder **java** dalam Project: Tampilan Android. Setiap nama modul diawali dengan nama domain (seperti **com.example.android**) dan menyertakan nama aplikasi.

Contoh berikut menampilkan komponen *aktivitas*:

1. Klik folder modul untuk meluaskan dan menampilkan file `MainActivity` untuk aktivitas yang ditulis di Java (kelas `MainActivity`).
2. Klik dua kali **MainActivity** untuk melihat file sumber di panel pengeditan, seperti yang ditampilkan dalam gambar di bawah ini.

```
package com.example.android.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Bagian paling atas file `MainActivity.java` adalah pernyataan `package` yang mendefinisikan paket aplikasi. Ini diikuti dengan blok `import` yang diringkas dalam gambar di atas, dengan " ... ". Klik titik untuk meluaskan blok guna menampilkannya. Pernyataan `import` akan mengimpor pustaka yang diperlukan untuk aplikasi, seperti berikut ini, yang mengimpor pustaka `AppCompatActivity`:

```
import android.support.v7.app.AppCompatActivity;
```

Setiap aktivitas dalam aplikasi diimplementasikan sebagai kelas Java. Deklarasi kelas berikut memperluas kelas `AppCompatActivity` untuk mengimplementasikan fitur dengan cara yang kompatibel-mundur dengan Android versi sebelumnya:

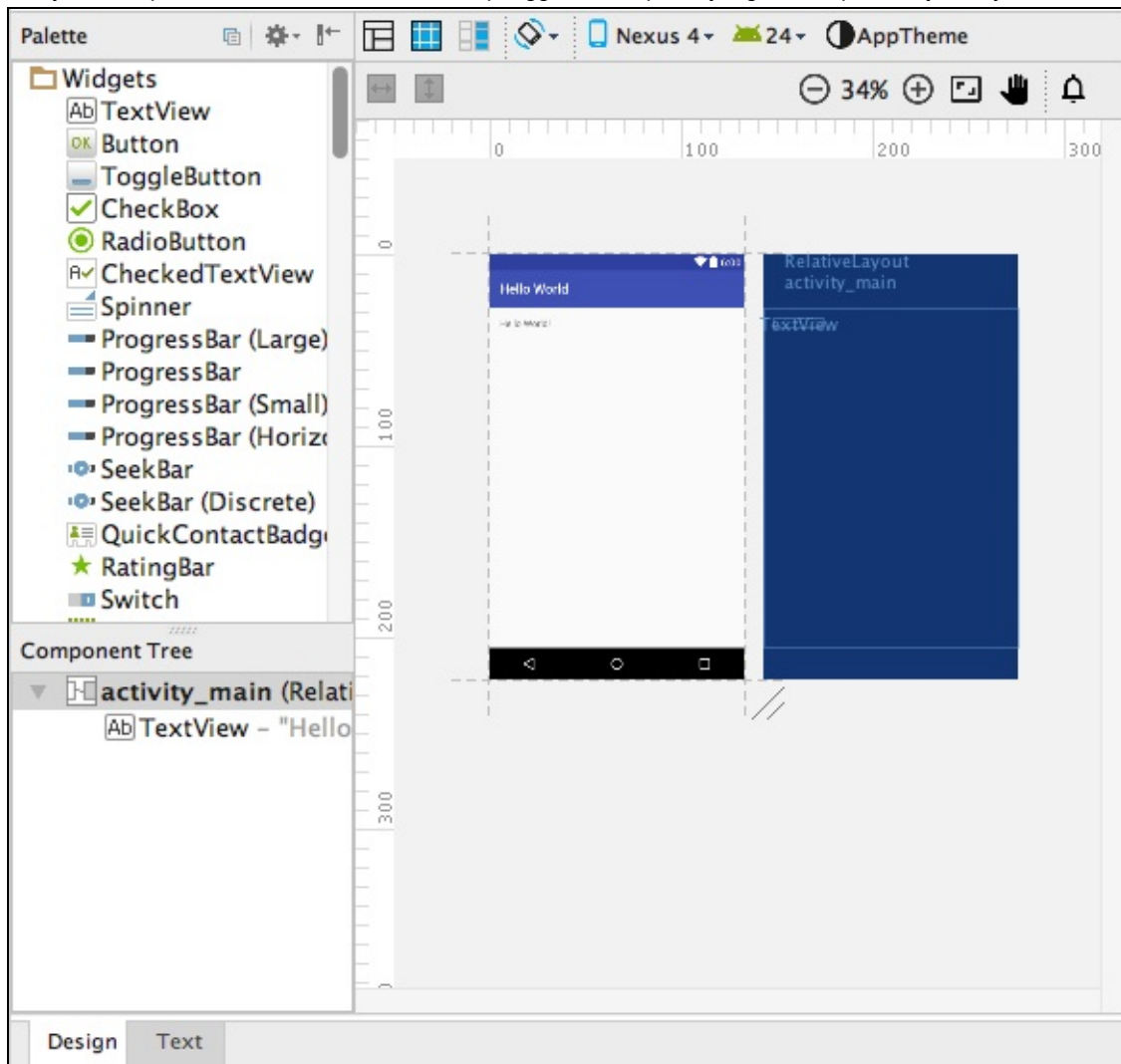
```
MainActivity kelas publik memperluas AppCompatActivity {
    ...
}
```

Seperti yang Anda pelajari terdahulu, sebelum sistem Android bisa memulai komponen aplikasi seperti aktivitas, sistem harus mengetahui bahwa aktivitas ada dengan membaca file `AndroidManifest.xml` aplikasi. Oleh karena itu, setiap aktivitas harus dicantumkan di file `AndroidManifest.xml`.

Menampilkan dan mengedit layout

Sumber daya layout ditulis dalam XML dan dicantumkan dalam folder **layout** di folder **res** dalam Project: Tampilan Android. Klik **res > layout**, kemudian klik dua kali **activity_main.xml** untuk melihat file layout di panel pengeditan.

Android Studio menampilkan tampilan Desain layout, seperti yang ditampilkan dalam gambar di bawah ini. Tampilan ini menyediakan panel Palette elemen antarmuka pengguna, dan petak yang menampilkan layout layar.



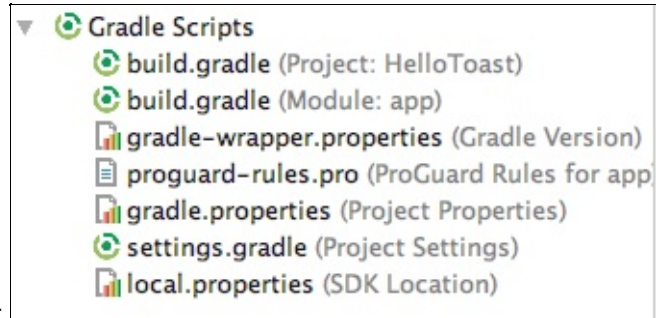
Memahami proses pembangunan

Android Application Package (APK) adalah format file paket untuk mendistribusikan dan memasang aplikasi seluler Android. Proses pembangunan melibatkan alat dan proses yang secara otomatis mengonversi setiap proyek menjadi APK.

Android Studio menggunakan Gradle sebagai dasar sistem pembangunan, dengan kemampuan khusus Android lebih banyak yang disediakan oleh Android Plugin for Gradle. Sistem pembangunan ini berjalan sebagai alat (bantu) terintegrasi dari menu Android Studio.

Memahami file build.gradle

Bila Anda membuat proyek, Android Studio secara otomatis menghasilkan file pembangunan penting di folder **Gradle Scripts** dalam Project: Tampilan Android. File pembangunan Android Studio diberi nama **build.gradle** seperti yang



ditampilkan di bawah ini:

Setiap proyek memiliki hal berikut:

build.gradle (Project: *apptitle*)

Ini adalah file pembangunan tingkat atas untuk keseluruhan proyek, berada di akar direktori proyek, yang mendefinisikan konfigurasi pembangunan yang berlaku untuk semua modul di proyek Anda. File ini, yang dihasilkan oleh Android Studio, tidak boleh diedit untuk menyertakan dependensi aplikasi.

build.gradle (Module: *app*)

Android Studio membuat file `build.gradle (Module: app)` terpisah untuk setiap modul. Anda bisa mengedit setelan pembangunan guna menyediakan opsi pemaketan khusus untuk setiap modul, seperti tipe pembangunan tambahan dan ragam produk, dan untuk menggantikan setelan di file manifest atau `build.gradle` tingkat atas. File ini paling sering adalah file untuk mengedit ketika mengubah konfigurasi tingkat aplikasi, seperti mendeklarasikan dependensi di bagian

`dependencies`. Yang berikut ini menampilkan isi file `build.gradle (Module: app)` proyek:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    buildToolsVersion "24.0.1"
    defaultConfig {
        applicationId "com.example.android.helloworld2"
        minSdkVersion 15
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.1'
    testCompile 'junit:junit:4.12'
}
```

File `build.gradle` menggunakan sintaks Gradle. Gradle adalah Domain Specific Language (DSL) untuk menjelaskan dan memanipulasi logika pembangunan dengan menggunakan [Groovy](#), yang merupakan bahasa dinamis untuk Java Virtual Machine (JVM). Anda tidak perlu mempelajari Groovy untuk melakukan perubahan, karena Android Plugin for Gradle

memperkenalkan sebagian besar elemen DSL yang Anda perlukan.

Tip: Untuk mengetahui selengkapnya tentang DSL plugin Android, baca [Dokumentasi referensi DSL](#).

Blok Plugin dan Android.

Dalam file `build.gradle (Module: app)` di atas, pernyataan pertama menerapkan tugas pembangunan plug-in Gradle khusus Android:

```
apply plugin: 'com.android.application'

android {
    ...
}
```

Blok `android { }` menetapkan hal berikut ini untuk versi:

- Versi SDK target untuk mengompilasi kode:

```
compileSdkVersion 24
```

- Versi alat pembangunan untuk digunakan membangun aplikasi:

```
buildToolsVersion "24.0.1"
```

Blok defaultConfig

Setelan inti dan entri untuk aplikasi ditetapkan di blok `defaultConfig { }` dalam `android { }` block:

```
...
defaultConfig {
    applicationId "com.example.hello.helloworld"
    minSdkVersion 15
    targetSdkVersion 23
    versionCode 1
    versionName "1.0"
    testInstrumentationRunner
        "android.support.test.runner.AndroidJUnitRunner"
}
...
```

Setelan `minSdkVersion` dan `targetSdkVersion` menggantikan setelan `AndroidManifest.xml` untuk versi SDK minimum dan versi SDK target. Lihat "Mendeklarasikan versi Android" sebelumnya di bagian ini untuk informasi latar belakang tentang setelan ini.

Pernyataan `testInstrumentationRunner` ini menambahkan dukungan instrumentasi untuk menguji antarmuka pengguna dengan Espresso dan UIAutomator. Hal ini dijelaskan dalam pelajaran terpisah.

Tipe pembangunan

Tipe pembangunan untuk aplikasi ditetapkan di blok `buildTypes { }`, yang mengontrol cara aplikasi dibangun dan dipaketkan.

```

...
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'),
            'proguard-rules.pro'
    }
}
...

```

Tipe pembangunan yang ditetapkan adalah `release` untuk rilis aplikasi. Tipe pembangunan umum lainnya adalah `debug`. Mengonfigurasi tipe pembangunan dijelaskan dalam pelajaran terpisah.

Dependensi

Dependensi untuk aplikasi didefinisikan di blok `dependencies { }`, yang merupakan bagian file `build.gradle` yang paling mungkin berubah saat Anda mulai mengembangkan kode yang bergantung pada pustaka. Blok adalah bagian dari Gradle API standar dan termasuk *di luar* blok `android { }`.

```

...
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:24.2.0'
    testCompile 'junit:junit:4.12'
}

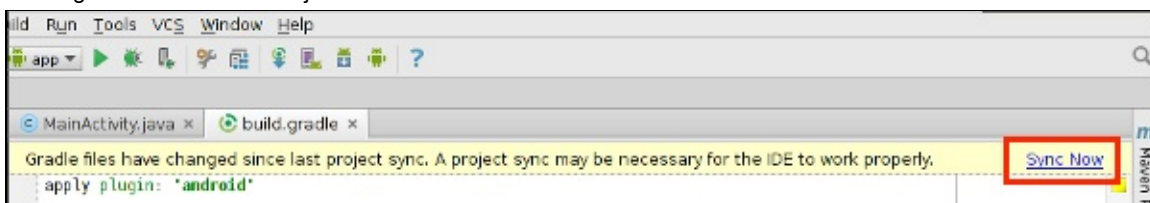
```

Dalam cuplikan di atas, pernyataan `compile fileTree(dir: 'libs', include: ['*.jar'])` menambahkan dependensi semua file ".jar" di dalam direktori `libs`. Konfigurasi `compile` akan mengompilasi aplikasi utama — segala sesuatu di dalamnya ditambahkan ke classpath kompilasi, dan juga dipaketkan ke dalam APK final.

Menyinkronkan proyek Anda

Bila Anda membuat perubahan pada file konfigurasi pembangunan dalam proyek, Android Studio akan mengharuskan Anda untuk melakukan *sinkronisasi* file proyek sehingga Android Studio bisa mengimpor perubahan konfigurasi pembangunan dan menjalankan beberapa pemeriksaan untuk memastikan konfigurasi tidak akan menimbulkan kesalahan pembangunan.

Untuk menyinkronkan file proyek, klik **Sync Now** di bilah notifikasi yang muncul saat membuat perubahan, atau klik **Sync Project** dari bilah menu. Jika Android Studio memperlihatkan kesalahan apa pun dengan konfigurasi — misalnya, jika kode sumber menggunakan fitur API yang hanya tersedia di API level yang lebih tinggi dari `compileSdkVersion` — jendela Messages muncul untuk menjelaskan masalah.



Menjalankan aplikasi pada emulator atau perangkat

Dengan emulator perangkat virtual, Anda bisa menguji aplikasi pada perangkat yang berbeda seperti tablet atau ponsel cerdas — dengan API level yang berbeda untuk versi Android yang berbeda — untuk memastikan aplikasi terlihat bagus dan berfungsi untuk sebagian besar pengguna. Meskipun ini adalah ide yang bagus, Anda tidak harus bergantung pada memiliki perangkat fisik yang tersedia untuk development aplikasi.

Pengelola Android Virtual Device (AVD) membuat perangkat virtual atau emulator yang menyimulasikan konfigurasi untuk tipe perangkat Android tertentu. Gunakan AVD Manager untuk mendefinisikan karakteristik perangkat keras perangkat dan API levelnya, dan untuk menyimpannya sebagai konfigurasi perangkat virtual. Ketika Anda memulai emulator Android, emulator akan membaca konfigurasi yang ditetapkan dan membuat perangkat emulasi pada komputer yang berperilaku sama persis dengan versi fisik perangkat.

Membuat perangkat virtual

Untuk menjalankan emulator pada komputer, gunakan AVD Manager untuk membuat konfigurasi yang menjelaskan

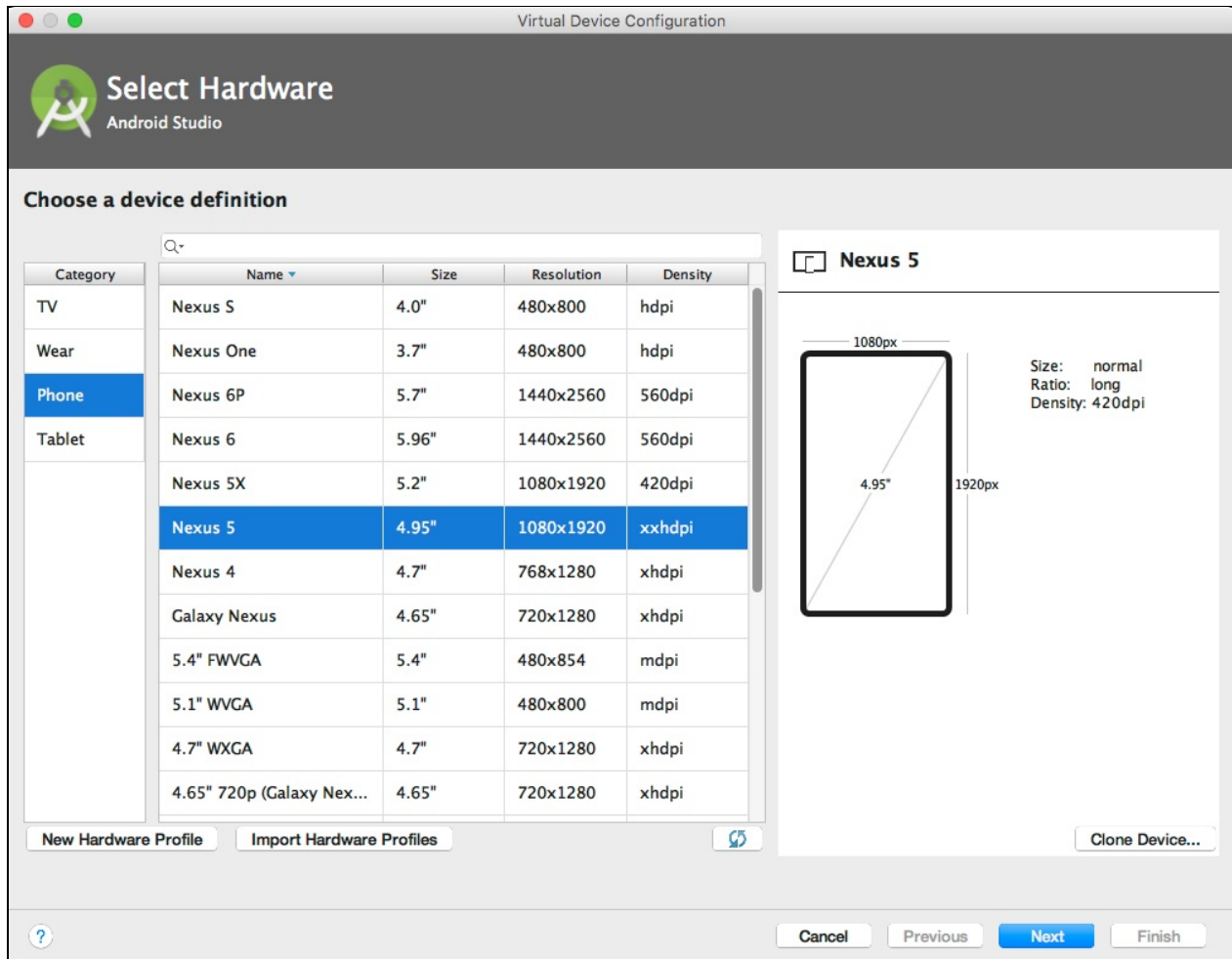
perangkat virtual. Pilih **Tools > Android > AVD Manager**, atau klik ikon AVD Manager  di bilah alat.

Layar "Your Virtual Devices" muncul menampilkan semua perangkat virtual yang dibuat sebelumnya. Klik tombol **+Create Virtual Device** untuk membuat perangkat virtual baru.



Anda bisa memilih perangkat dari daftar perangkat keras yang telah didefinisikan sebelumnya. Untuk setiap perangkat, tabel menampilkan ukuran tampilan diagonal (Size), resolusi layar dalam piksel (Resolution), dan kepadatan piksel (Density). Misalnya, kepadatan piksel perangkat Nexus 5 adalah `xxhdpi`, artinya aplikasi menggunakan ikon di folder


xxhdpi dari folder **mipmap**. Selain itu, aplikasi akan menggunakan layout dan sumber daya dapat digambar dari folder yang didefinisikan untuk kepadatan tersebut.



Anda juga memilih versi sistem Android untuk perangkat. Tab **Recommended** menampilkan sistem yang disarankan untuk perangkat. Lebih banyak versi tersedia di dalam tab **x86 Images** dan **Other Images**.

Menjalankan aplikasi pada perangkat virtual

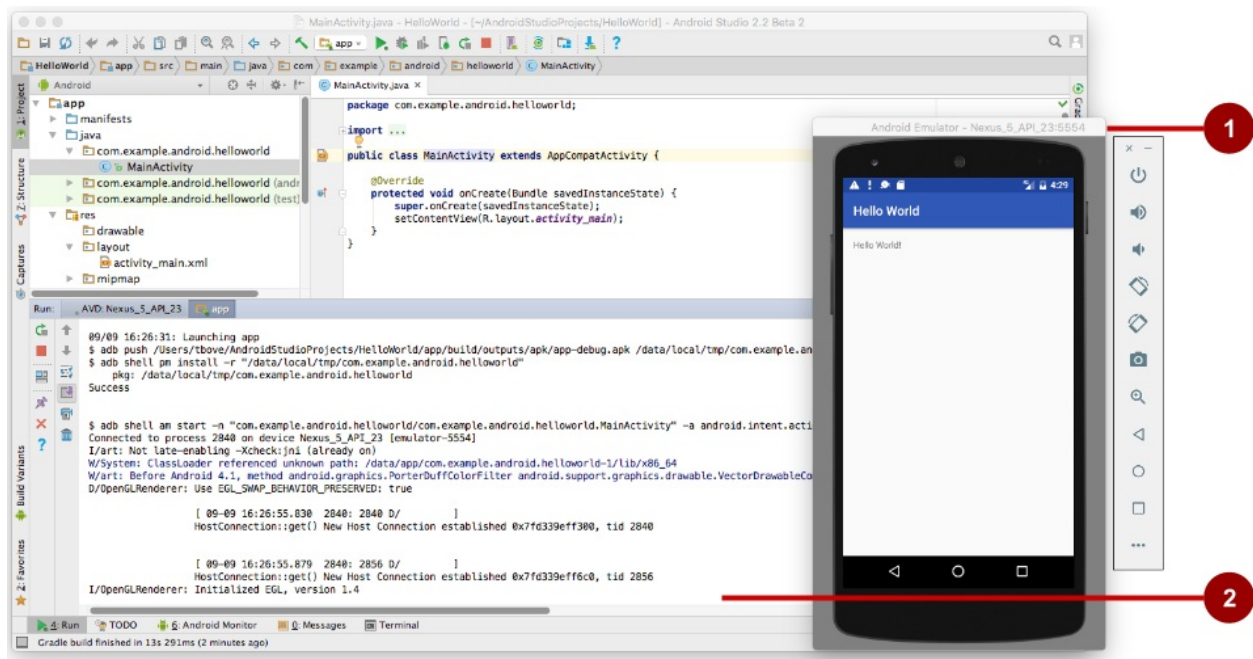
Untuk menjalankan aplikasi pada perangkat virtual yang Anda buat di bagian sebelumnya, ikuti langkah-langkah ini:

1. Di Android Studio, pilih **Run > Run app** atau klik **Run icon**  di bilah alat.
2. Di jendela Select Deployment Target, pada Emulator yang tersedia, pilih perangkat virtual yang Anda buat, dan klik **OK**.

Emulator memulai dan mem-booting seperti perangkat fisik. Ini memerlukan waktu beberapa saat, bergantung pada kecepatan komputer Anda. Aplikasi membangun, dan setelah emulator siap, Android Studio mengunggah aplikasi ke emulator dan menjalankannya.

Anda harus melihat aplikasi yang dibuat dari template Empty Activity ("Hello World") seperti yang ditampilkan dalam gambar berikut ini, yang juga menampilkan panel Run Android Studio yang menampilkan tindakan yang dilakukan untuk menjalankan aplikasi di emulator.

Catatan: Saat menguji pada emulator, praktik yang baik adalah memulainya sekali pada awal sesi, dan jangan menutupnya hingga selesai sehingga tidak perlu melalui proses booting kembali.



Dalam gambar di atas:

1. **Emulator** menjalankan aplikasi.
2. **Run Pane**. Ini menampilkan tindakan yang dilakukan untuk memasang dan menjalankan aplikasi.

Menjalankan aplikasi pada perangkat fisik

Selalu uji aplikasi Anda pada perangkat fisik, karena pengguna akan menggunakan aplikasi pada perangkat fisik. Meskipun emulator cukup bagus, emulator tidak bisa menampilkan semua kemungkinan keadaan perangkat, seperti yang terjadi jika ada panggilan masuk sewaktu aplikasi berjalan. Untuk menjalankan aplikasi pada perangkat fisik, Anda memerlukan hal berikut ini:

- Perangkat Android seperti ponsel cerdas atau tablet.
- Kabel data untuk menghubungkan perangkat Android ke komputer melalui porta USB.
- Jika Anda menggunakan Linux atau Windows, mungkin perlu melakukan langkah tambahan untuk menjalankan aplikasi pada perangkat keras. Periksa dokumentasi [Menggunakan Perangkat Keras](#). Pada Windows, Anda mungkin perlu memasang driver USB yang sesuai untuk perangkat. Lihat [Driver USB OEM](#).

Untuk memungkinkan Android Studio berkomunikasi dengan perangkat, aktifkan USB Debugging pada perangkat Android. Pada versi Android 4.2 dan yang lebih baru, layar opsi Developer disembunyikan secara default. Ikuti langkah-langkah ini untuk mengaktifkan USB Debugging:

1. Pada perangkat fisik, buka **Settings** dan pilih **About phone** di bagian bawah layar Settings.
2. Ketuk informasi **Build number** tujuh kali.

Anda membacanya dengan benar: Ketuk *tujuh kali*.

3. Kembali ke layar sebelumnya (**Settings**). **Developer Options** saat ini muncul di bagian bawah layar. Ketuk **Developer options**.
4. Pilih **USB Debugging**.

Sekarang, hubungkan perangkat dan jalankan aplikasi dari Android Studio.

Pemecahan masalah koneksi perangkat

Jika Android Studio tidak mengenali perangkat, coba hal berikut:

1. Putuskan koneksi perangkat dari komputer, kemudian hubungkan kembali.

2. Mulai ulang Android Studio.
3. Jika komputer masih tidak menemukan perangkat atau mendeklarasikannya "tidak sah":
 - i. Putuskan koneksi perangkat dari komputer.
 - ii. Di perangkat, pilih **Settings > Developer Options**.
 - iii. Ketuk **Revoke USB Debugging authorizations**.
 - iv. Hubungkan kembali perangkat ke komputer.
 - v. Bila dikonfirmasi, berikan otorisasi.
4. Anda mungkin perlu memasang driver USB yang sesuai untuk perangkat. Lihat [dokumentasi Menggunakan Perangkat Keras](#).
5. Periksa dokumentasi terakhir, forum pemrograman, atau dapatkan bantuan dari instruktur Anda.

Menggunakan log

Log merupakan alat (bantu) debug andal yang bisa Anda gunakan untuk melihat nilai, jalur eksekusi, dan pengecualian. Setelah menambahkan pernyataan log ke aplikasi, pesan log Anda akan muncul bersama pesan log umum di tab **logcat** panel Android Monitor di Android Studio.

Untuk melihat panel Android Monitor, klik tombol **Android Monitor** di bagian bawah jendela utama Android Studio. Android Monitor menawarkan dua tab:

- Tab **logcat**. Tab **logcat** menampilkan pesan log tentang aplikasi saat aplikasi berjalan. Jika menambahkan pernyataan logging ke aplikasi, pesan log Anda dari pernyataan tersebut muncul dengan pesan log lain di bawah tab ini.
- Tab **Monitors**. Tab **Monitors** memantau kinerja aplikasi, yang bisa digunakan untuk men-debug dan menyesuaikan kode Anda.

Menambahkan pernyataan log ke aplikasi Anda

Pernyataan log menambahkan pesan apa pun yang Anda tetapkan ke log. Menambahkan pernyataan log di titik tertentu dalam kode memungkinkan developer melihat nilai, jalur eksekusi, dan pengecualian.

Misalnya, pernyataan log berikut menambahkan `"MainActivity"` dan `"Hello World"` ke log:

```
Log.d("MainActivity", "Hello World");
```

Berikut adalah elemen pernyataan ini:

- `Log` : Kelas [Log](#) adalah API untuk mengirim pesan log.
- `d` : Anda menetapkan *log level* sehingga Anda bisa memfilter pesan log menggunakan menu tarik-turun di bagian tengah panel tab **logcat**. Berikut adalah tingkat log yang bisa Anda tetapkan:
 - `d` : Pilih **Debug** atau **Verbose** untuk melihat pesan ini.
 - `e` : Pilih **Error** atau **Verbose** untuk melihat pesan ini.
 - `w` : Pilih **Warning** atau **Verbose** untuk melihat pesan ini.
 - `i` : Pilih **Info** atau **Verbose** untuk melihat pesan ini.
- `"MainActivity"` : Argumen pertama adalah *log tag* yang bisa digunakan untuk memfilter pesan pada tab **logcat**. Ini biasanya merupakan nama aktivitas tempat pesan berasal. Akan tetapi, Anda bisa membuatnya menjadi apa saja yang berguna bagi Anda untuk men-debug aplikasi. Praktik terbaik adalah untuk menggunakan konstanta sebagai tag log, seperti berikut:

1. Definisikan tag log sebagai konstanta sebelum menggunakannya dalam pernyataan log:

```
private static final String LOG_TAG =  
    MainActivity.class.getSimpleName();
```

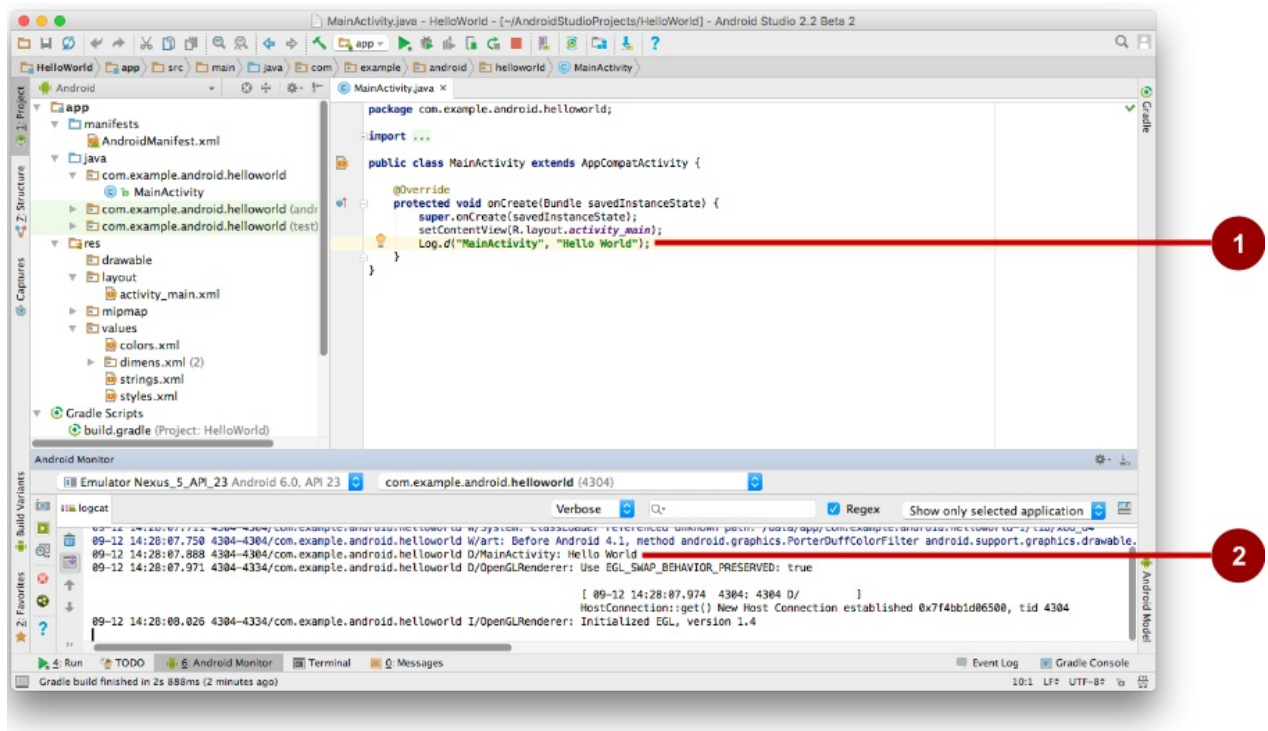

- Gunakan konstanta dalam pernyataan log:

```
Log.d(LOG_TAG, "Hello World");
```

- "Hello World" : Argumen kedua adalah pesan sebenarnya yang muncul setelah log level dan tag log pada tab **logcat**.

Menampilkan pesan log Anda

Panel Run muncul di tempat panel Android Monitor bila Anda menjalankan aplikasi di emulator atau perangkat. Setelah mulai menjalankan aplikasi, klik tombol **Android Monitor** di bagian bawah jendela utama, kemudian klik tab **logcat** di panel Android Monitor jika belum dipilih.



Dalam gambar di atas:

- Pernyataan log di metode `onCreate()` dari `MainActivity`.
- Panel Android Monitor yang menampilkan pesan log **logcat**, termasuk pesan dari pernyataan log.

Secara default, tampilan log disetel ke **Verbose** di menu tarik-turun di bagian atas tampilan **logcat** untuk menampilkan semua pesan. Anda bisa mengubahnya menjadi **Debug** untuk melihat pesan yang dimulai dengan `Log.d`, atau mengubahnya menjadi **Error** untuk melihat pesan yang dimulai dengan `Log.e`, dan seterusnya.