19CSI605 - Mobile Application Development

Building your first app

G. Pradeep, AP

Introduction to Android

What is Android?

- **Android** is an operating system and programming platform developed by Google for smartphones and other mobile devices (such as tablets).
- It can run on many different devices from many different manufacturers.
- Android includes a software development kit for writing original code and assembling software modules to create apps for Android users.
- It also provides a marketplace to distribute apps.
- All together, Android represents an ecosystem for mobile apps.

Introduction to Android

Example

Introduction to Android

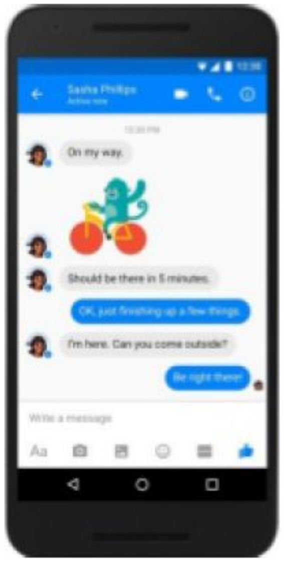
Why develop apps for Android?

- Apps are developed for a variety of reasons: addressing business requirements, building new services, creating new businesses, and providing games and other types of content for users.
- Developers choose to develop for Android in order to reach the majority of mobile device users.

Introduction to Android

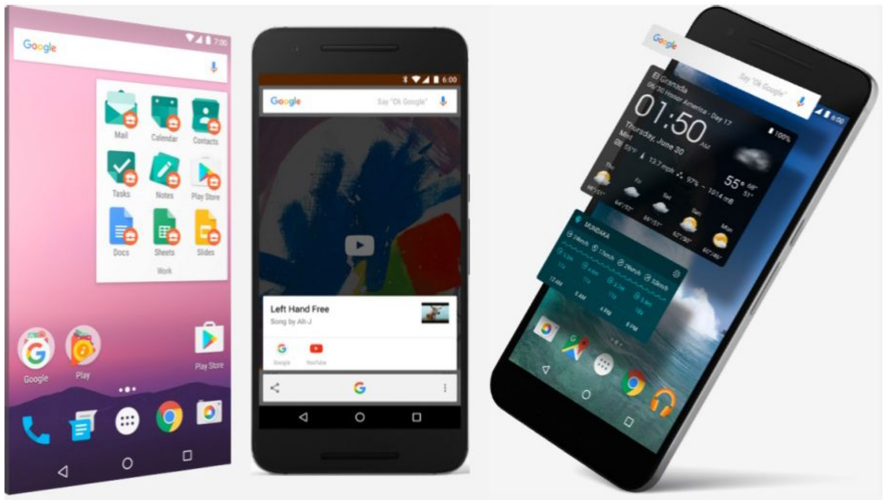
Most popular platform for mobile apps

- Android powers hundreds of millions of mobile devices in more than 190 countries around the world.
- It has the largest installed base of any mobile platform and is still growing fast.


Sasha Phillips
On my way.
Should be there in 5 minutes.
OK, just finishing up a few things
I'm here. Can you come outside?
Be right there!
Write a message

Best experience for app users

- Android provides a touch-screen user interface (UI) for interacting with apps.
- Android's user interface is mainly based on direct manipulation, using touch gestures such as swiping, tapping and pinching to manipulate on-screen objects



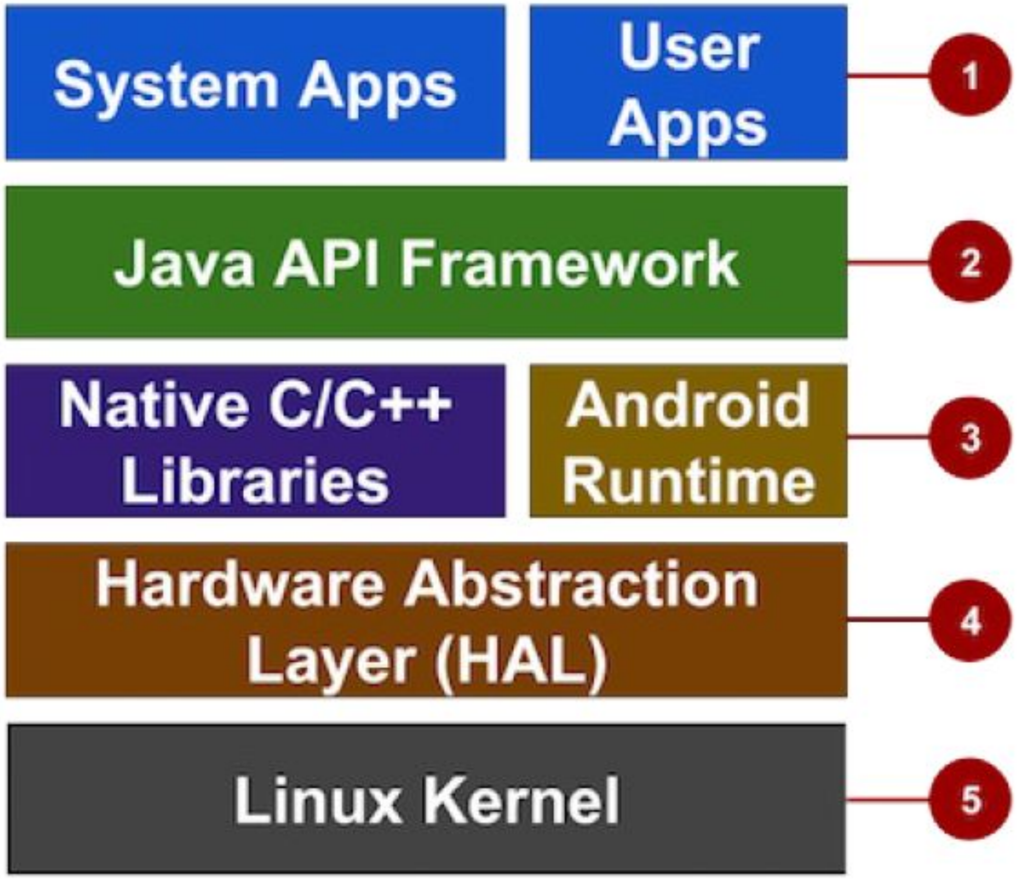
Introduction to Android

Easy to develop apps

- Use the Android software development kit (SDK) to develop apps that take advantage of the Android operating system and UI.
- With SDK, Use Java for developing the app and Extensible Markup Language (XML) files for describing data resources.
- To help you develop your apps efficiently, Google offers a full Java Integrated Development Environment (IDE) called Android Studio.
- Using Android Studio, you can develop on any available Android device, or create virtual devices that emulate any hardware configuration.

Introduction to Android

Android Architect



Android Architecture

1. **Apps:** Your apps live at this level, along with core system apps for email, SMS messaging, calendars, Internet browsing, or contacts.
2. **Java API Framework:** All features of Android are available to developers through application programming interfaces (APIs) written in the Java language. You don't need to know the details of all of the APIs to learn how to develop Android apps, but you can learn more about the following APIs, which are useful for creating apps:

- View System
- Resource Manager
- Notification Manager
- Activity Manager
- Content Providers

Android Architecture

3. Libraries and Android Runtime:

- ✓ Each app runs in its own process and with its own instance of the Android Runtime, which enables multiple virtual machines on low-memory devices.
- ✓ Android also includes a set of core runtime libraries that provide most of the functionality of the Java programming language, including some Java 8 language features that the Java API framework uses.
- ✓ Many core Android system components and services are built from native code that require native libraries written in C and C++.
- ✓ These native libraries are available to apps through the Java API framework.

Android Architecture

4. **Hardware Abstraction Layer (HAL):**

- ✓ This layer provides standard interfaces that expose device hardware capabilities to the higher-level Java API framework.
- ✓ The HAL consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or bluetooth module.

5. **Linux Kernel:**

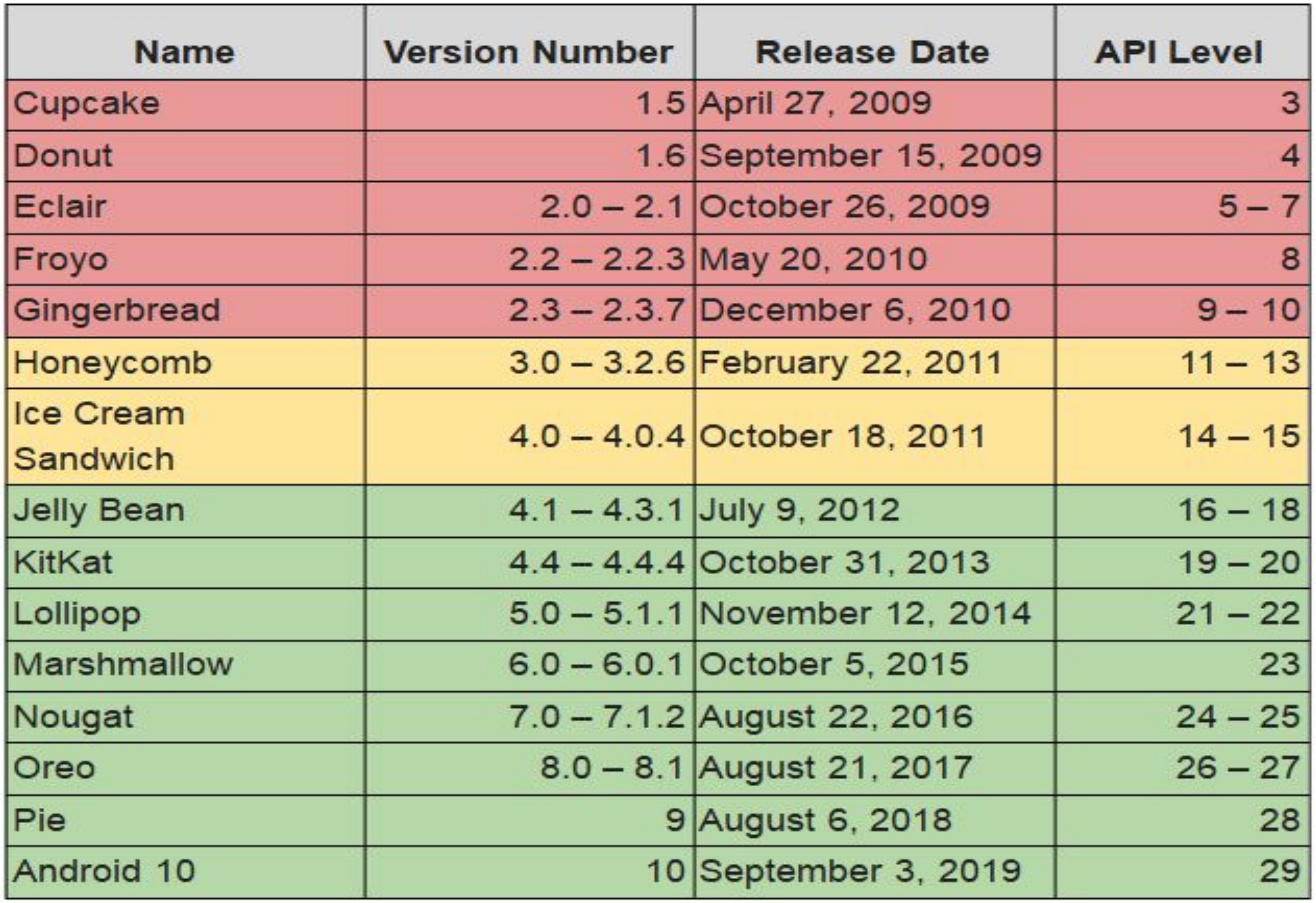
- ✓ The foundation of the Android platform is the Linux kernel. The above layers rely on the Linux kernel for underlying functionalities such as threading and low-level memory management.
- ✓ Using a Linux kernel enables Android to take advantage of key security features and allows device manufacturers to develop hardware drivers for a wellknown kernel.

Introduction to Android

Many distribution options

- You can distribute your Android app in many different ways: email, website or an app marketplace such as Google Play.
- Android users download billions of apps and games from the Google Play store each month

Android Versions



Name	Version Number	Release Date	API Level
Cupcake	1.5	April 27, 2009	3
Donut	1.6	September 15, 2009	4
Eclair	2.0 – 2.1	October 26, 2009	5 – 7
Froyo	2.2 – 2.2.3	May 20, 2010	8
Gingerbread	2.3 – 2.3.7	December 6, 2010	9 – 10
Honeycomb	3.0 – 3.2.6	February 22, 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	16 – 18
KitKat	4.4 – 4.4.4	October 31, 2013	19 – 20
Lollipop	5.0 – 5.1.1	November 12, 2014	21 – 22
Marshmallow	6.0 – 6.0.1	October 5, 2015	23
Nougat	7.0 – 7.1.2	August 22, 2016	24 – 25
Oreo	8.0 – 8.1	August 21, 2017	26 – 27
Pie	9	August 6, 2018	28
Android 10	10	September 3, 2019	29

Introduction to Android

The challenges of Android app development

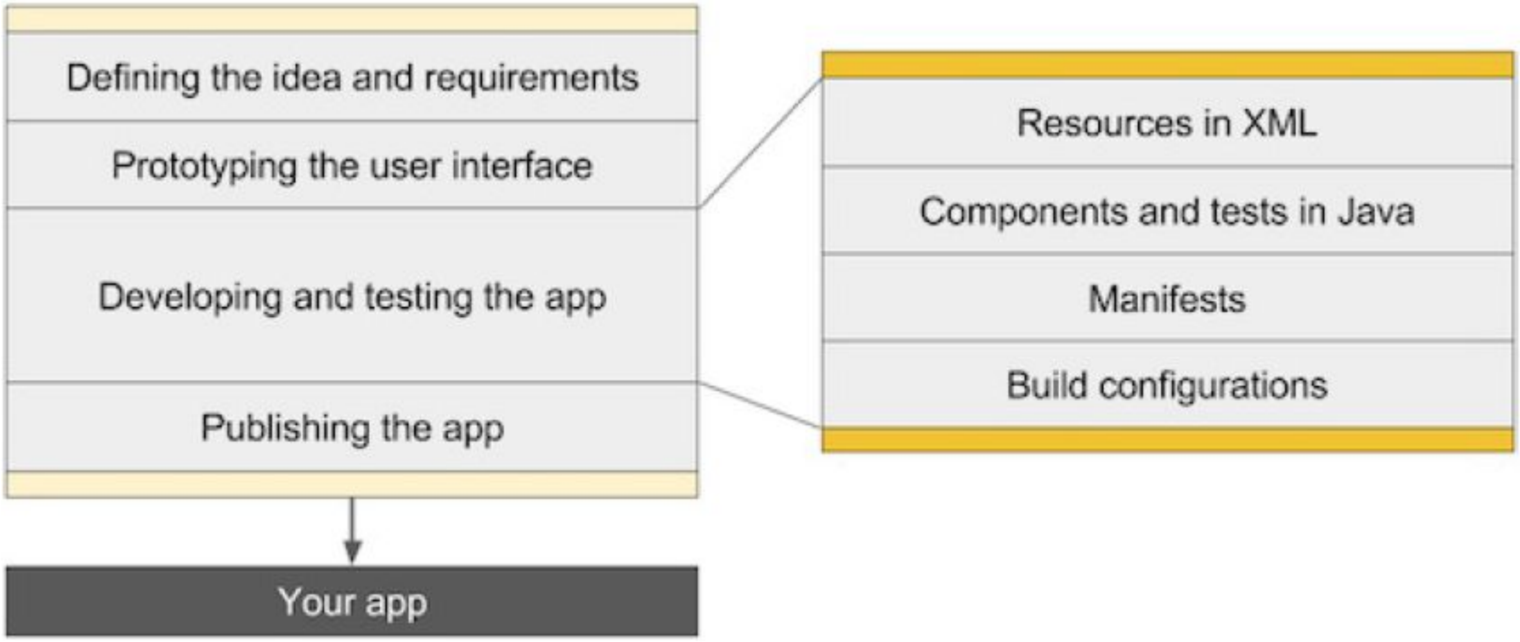
While the Android platform provide rich functionality for app development, there are still a number of challenges you need to address, such as:

- Building for a multi-screen world
- Getting performance right
- Keeping your code and your users secure
- Remaining compatible with older platform versions
- Understanding the market and the user

Create Your First Android App

The development process

- An Android app project begins with an idea and a definition of the requirements necessary to realize that idea.
- As the project progresses, it goes through design, development, and testing.


Defining the idea and requirements
Prototyping the user interface
Developing and testing the app
Publishing the app
Your app
Resources in XML
Components and tests in Java
Manifests
Build configurations

Using Android Studio

- Android Studio provides tools for the testing, and publishing phases of the development process, and a unified development environment for creating apps for all Android devices.
- The development environment includes code templates with sample code for common app features, extensive testing tools and frameworks, and a flexible build system.

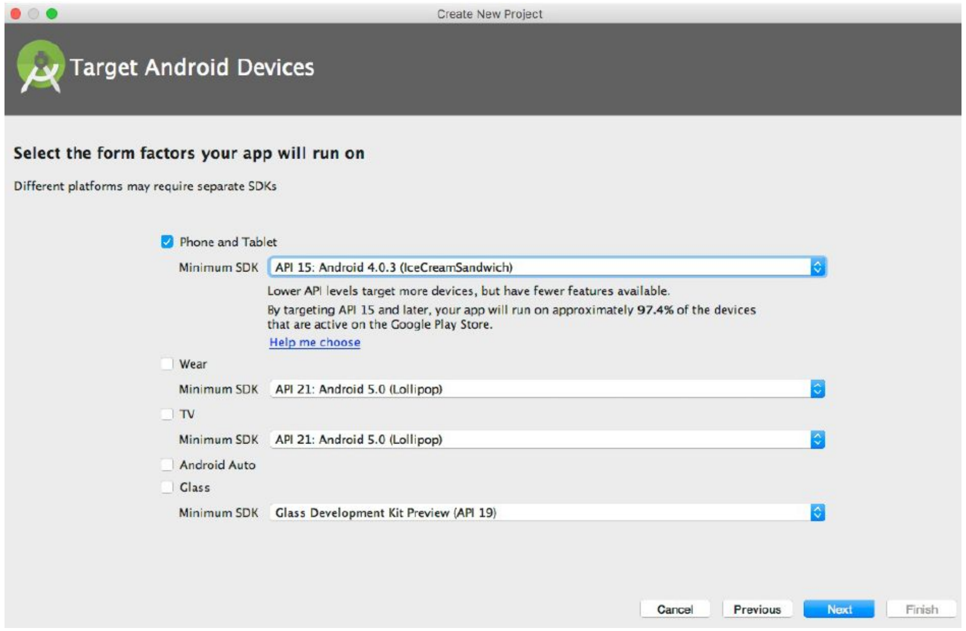
Starting an Android Studio project

- After you have successfully installed the Android Studio IDE, double-click the Android Studio application icon to start it.
- Choose Start a new Android Studio project in the Welcome window, and name the project the same name that you want to use for the app.

Using Android Studio

Choosing target devices and the minimum SDK

- When choosing Target Android Devices, Phone and Tablet are selected by default, as shown in the figure below. The choice shown in the figure for the Minimum SDK — **API 15: Android 4.0.3 (IceCreamSandwich) — makes your app** compatible with 97% of Android devices active on the Google Play Store



Select the form factors your app will run on

Different platforms may require separate SDKs

☑ Phone and Tablet
Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.
Help me choose

☐ Wear
Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV
Minimum SDK: API 21: Android 5.0 (Lollipop)

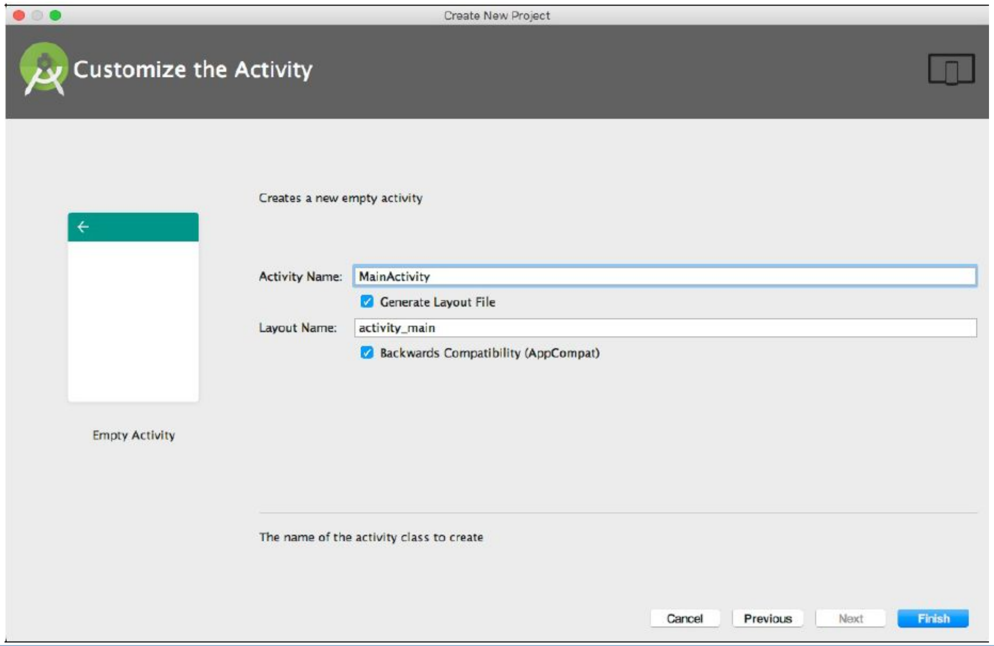
☐ Android Auto

☐ Glass
Minimum SDK: Glass Development Kit Preview (API 19)

Using Android Studio

Choosing a template

- Android Studio pre-populates your project with minimal code for an activity and a screen layout based on a *template.*
- *A* variety of templates are available, ranging from a virtually blank template (Add No Activity) to various types of activities.



Create New Project

Customize the Activity

Creates a new empty activity

Activity Name: MainActivity

☑ Generate Layout File

Layout Name: activity_main

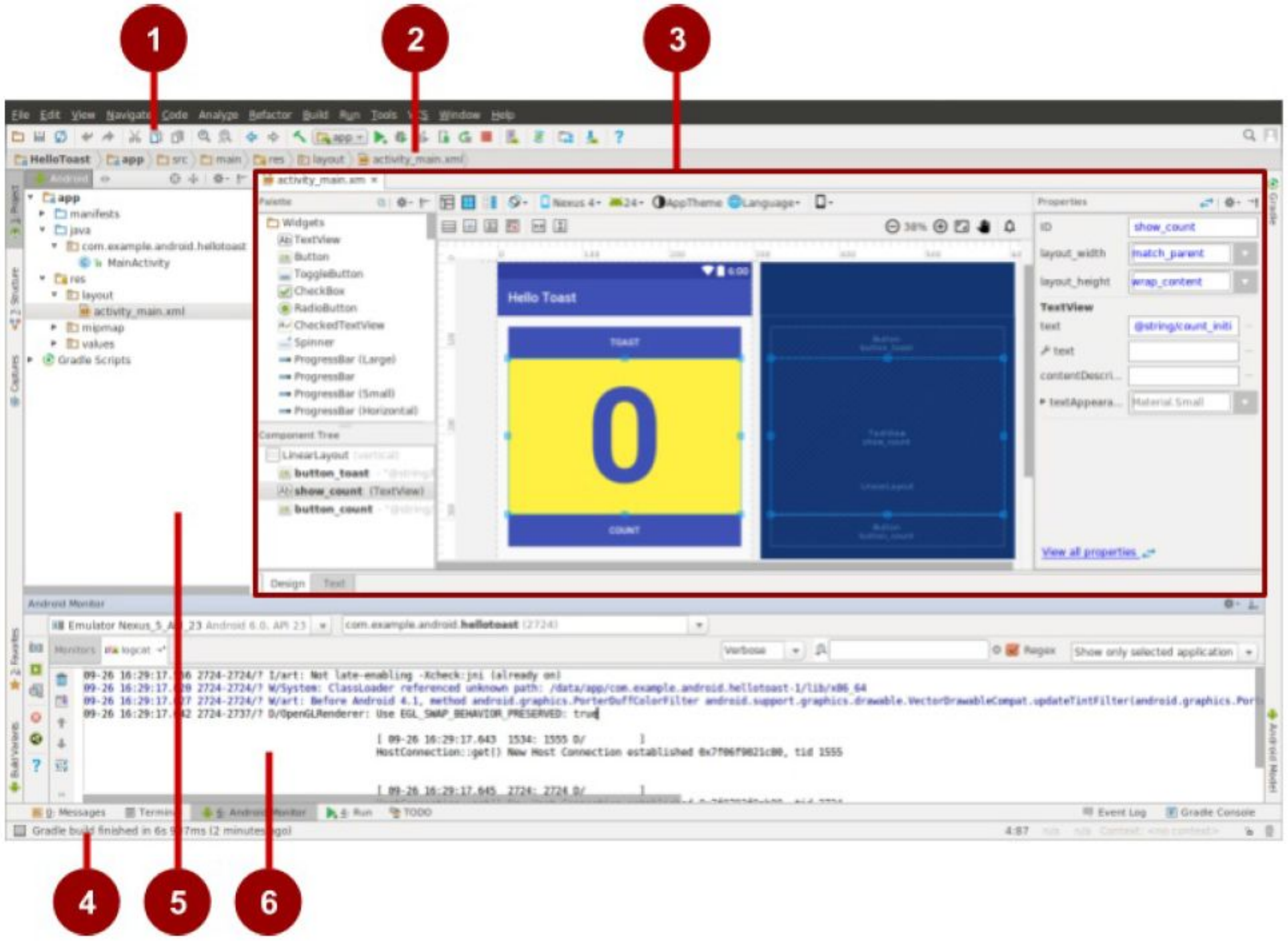
☑ Backwards Compatibility (AppCompat)

Empty Activity

The name of the activity class to create

Using Android Studio

Android Studio window panes

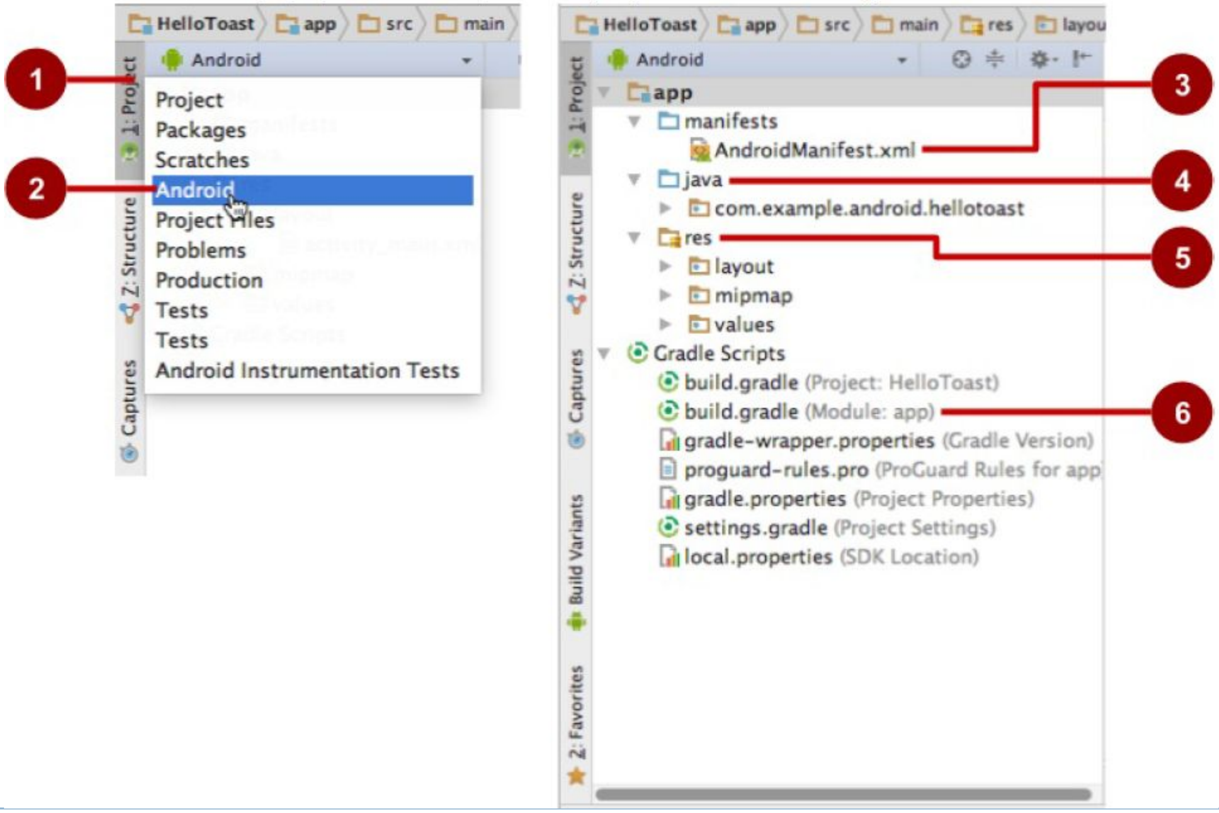


1. Toolbar
2. Navigation Bar
3. Editor Pane
4. Status Bar
5. Project Pane
6. Monitor Pane

Using Android Studio

Exploring a project

- Each project in Android Studio contains the AndroidManifest.xml file, component source-code files, and associated resource files


HelloToast
app
src
main
res
Android
1: Project
Z: Structure
Captures
Build Variants
2: Favorites
Project
Packages
Scratches
Android
Project Files
Problems
Production
Tests
Tests
Android Instrumentation Tests
app
manifests
AndroidManifest.xml
java
com.example.android.hellotoast
res
layout
mipmap
values
Gradle Scripts
build.gradle (Project: HelloToast)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
1
2
3
4
5
6

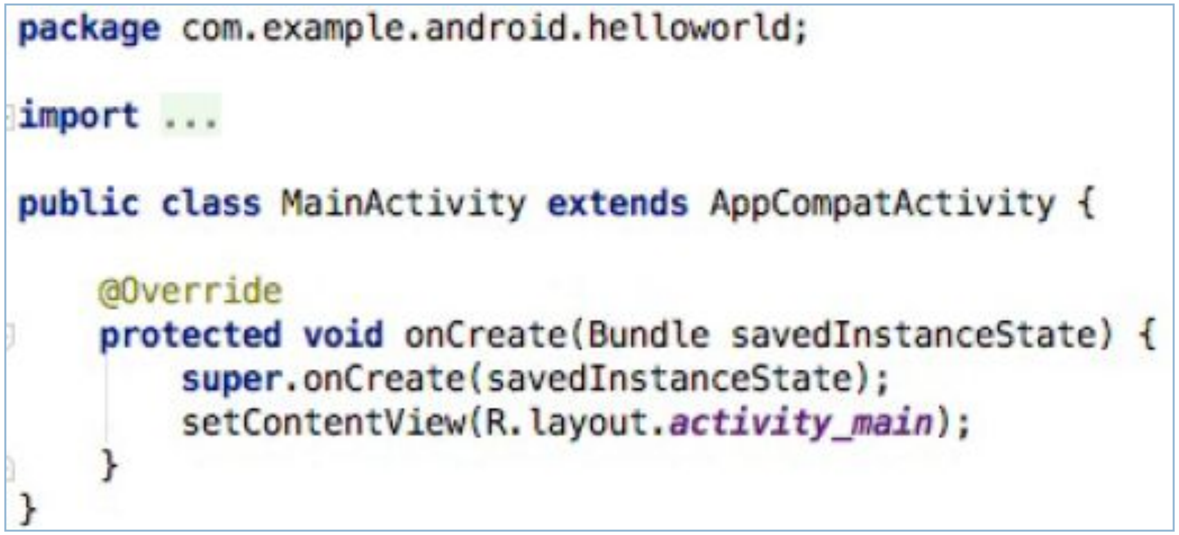
1. Project Tab
2. Android select
3. XML file
4. Java folder
5. res folder
6. Build.gradle

Using Android Studio

Viewing and editing Java code

The following example shows an *activity component:*

1. Click the module folder to expand it and show the MainActivity file for the activity written in Java (the MainActivity class).
2. Double-click **MainActivity** to see the source file in the editing pane, as shown in the figure below.

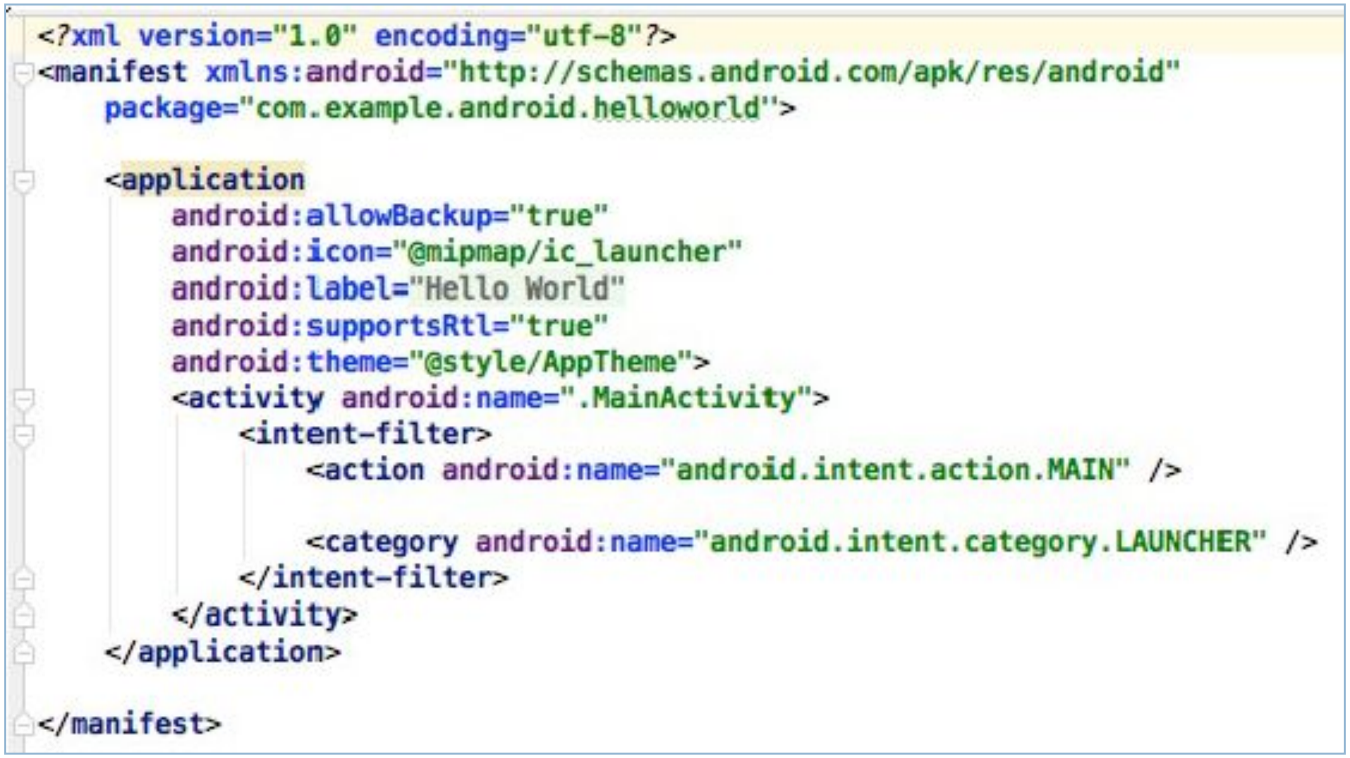


```
package com.example.android.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

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

Using Android Studio

Viewing the Android Manifest

To view this file, expand the manifests folder in the Project: Android view, and double-click the file (**AndroidManifest.xml).**

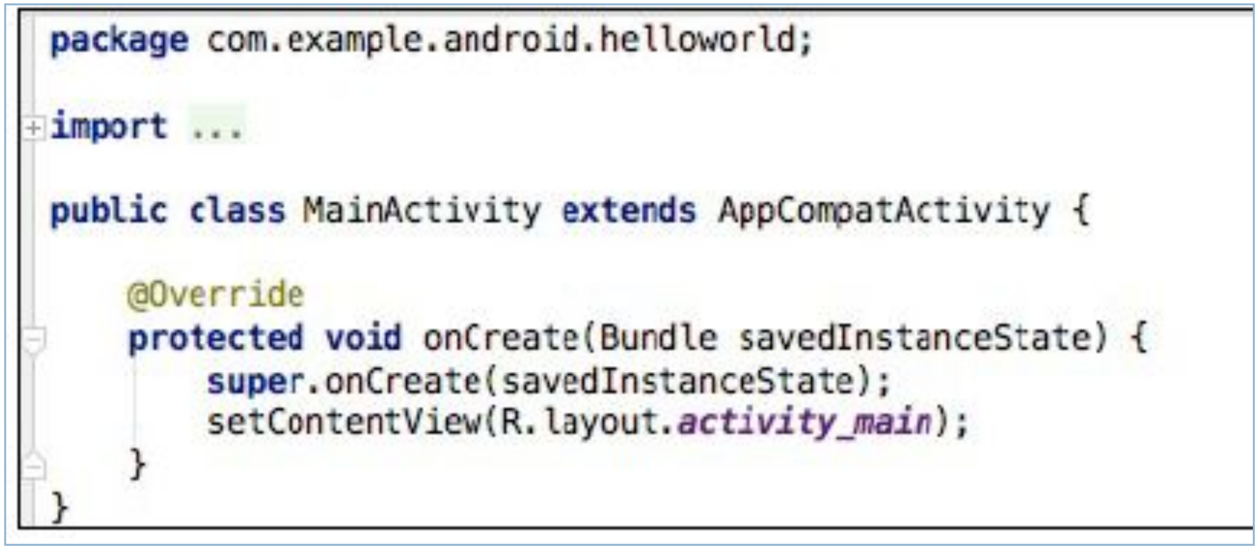


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.helloworld">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hello World"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Using Android Studio

Viewing and editing Java code

1. Click the module folder to expand it and show the MainActivity file for the activity written in Java (the MainActivity class).
2. Double-click MainActivity to see the source file in the editing pane, as shown in the figure below.

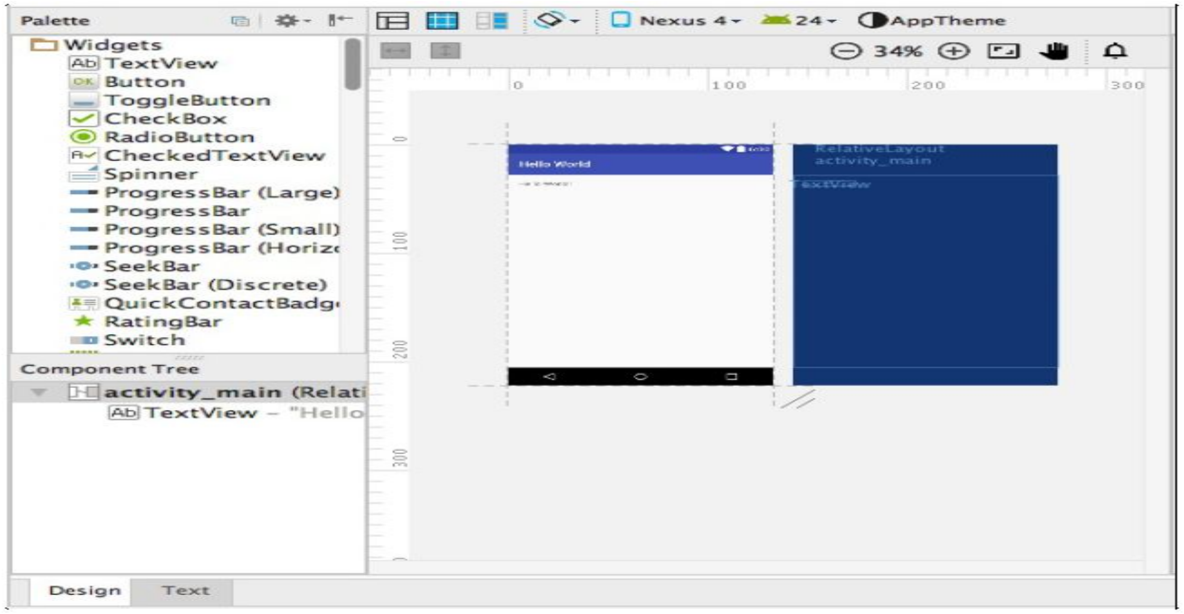


```
package com.example.android.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

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

Using Android Studio

Viewing and editing layouts

- Layout resources are written in XML and listed within the layout folder in the res folder in the Project: Android view.
- Click res > layout and then double-click activity_main.xml to see the layout file in the editing pane.



Using Android Studio

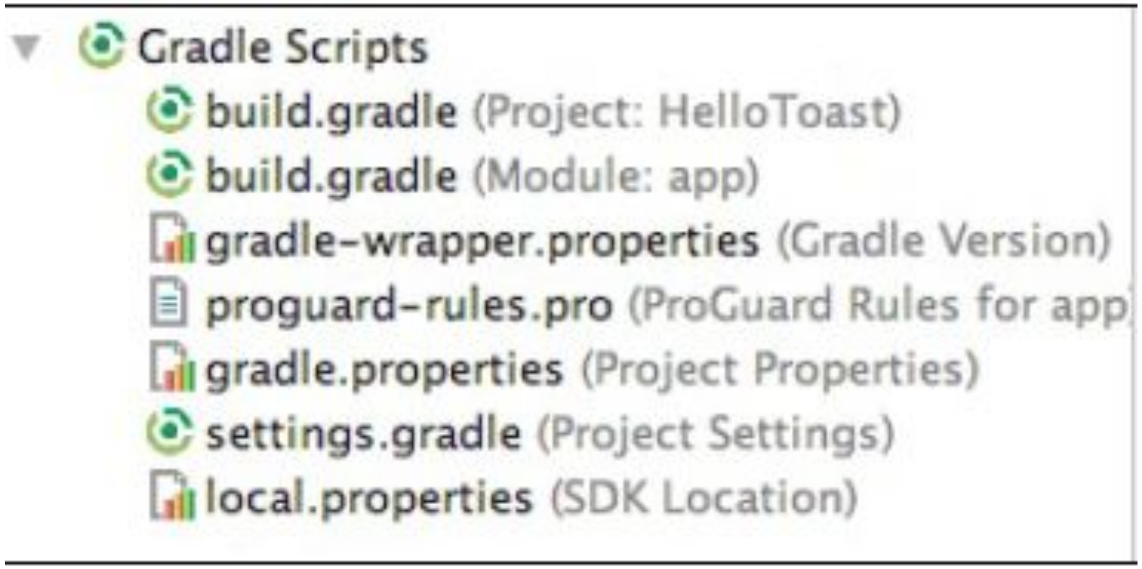
Understanding the build process

- The Android application package (APK) is the package file format for distributing and installing Android mobile apps.
- The build process involves tools and processes that automatically convert each project into an APK.
- Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android Plugin for Gradle.
- This build system runs as an integrated tool from the Android Studio menu.

Using Android Studio

Understanding the build process

- When you create a project, Android Studio automatically generates the necessary build files in the Gradle Scripts folder in Project: Android view. Android Studio build files are named build.gradle as shown below:



Gradle Scripts
build.gradle (Project: HelloToast)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

Using Android Studio

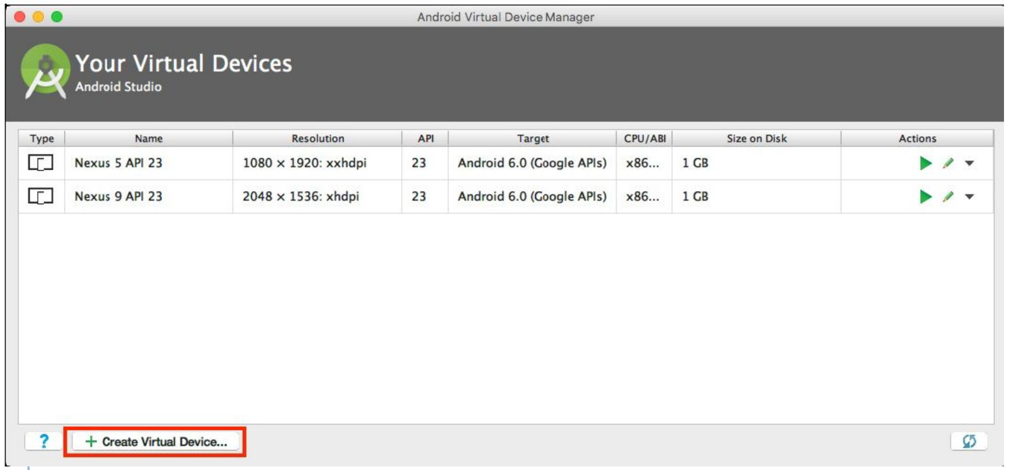
Running the app on an emulator or a device

- With virtual device emulators, you can test an app on different devices such as tablets or smartphones — with different API levels for different Android versions.
- The Android Virtual Device (AVD) manager creates a virtual device or emulator that simulates the configuration for a particular type of Android device.
- Use the AVD Manager to define the hardware characteristics of a device and its API level, and to save it as a virtual device configuration.
- When you start the Android emulator, it reads a specified configuration and creates an emulated device on your computer that behaves exactly like a physical version of that device

Using Android Studio

Creating a virtual device

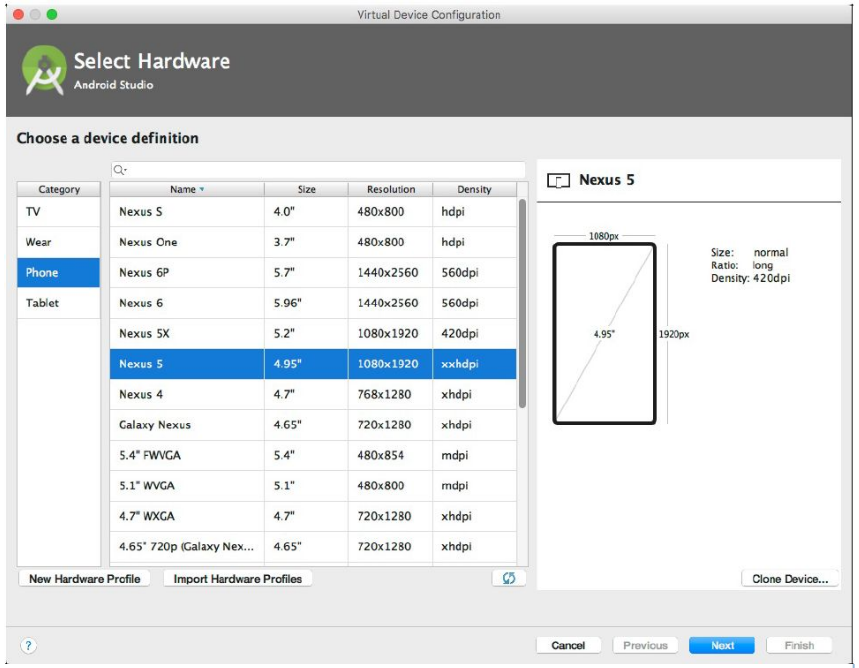
Select Tools > Android > AVD Manager, or click the AVD Manager icon in the toolbar



Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5 API 23	1080 × 1920: xxhdpi	23	Android 6.0 (Google APIs)	x86...	1 GB	
	Nexus 9 API 23	2048 × 1536: xhdpi	23	Android 6.0 (Google APIs)	x86...	1 GB	



Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

Category
TV
Wear
Phone
Tablet

Name	Size	Resolution	Density
Nexus S	4.0"	480x800	hdpi
Nexus One	3.7"	480x800	hdpi
Nexus 6P	5.7"	1440x2560	560dpi
Nexus 6	5.96"	1440x2560	560dpi
Nexus 5X	5.2"	1080x1920	420dpi
Nexus 5	4.95"	1080x1920	xxhdpi
Nexus 4	4.7"	768x1280	xhdpi
Galaxy Nexus	4.65"	720x1280	xhdpi
5.4" FWVGA	5.4"	480x854	mdpi
5.1" WVGA	5.1"	480x800	mdpi
4.7" WXGA	4.7"	720x1280	xhdpi
4.65" 720p (Galaxy Nex...	4.65"	720x1280	xhdpi

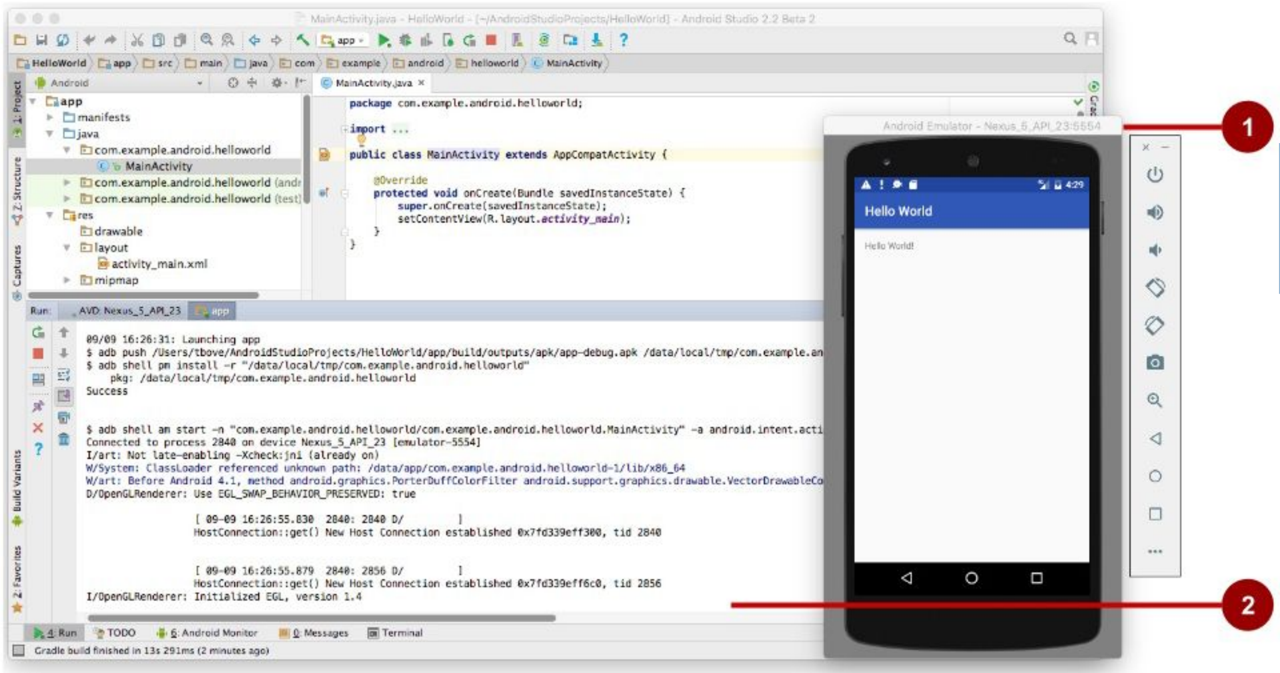
Nexus 5

Size: normal
Ratio: long
Density: 420dpi

Using Android Studio

Running the app on the virtual device

1. In Android Studio, select **Run > Run app or click the Run icon in the toolbar.**
2. In the Select Deployment Target window, under Available Emulators, select the virtual device you created, and click **OK.**


MainActivity.java - HelloWorld - [~/AndroidStudioProjects/HelloWorld] - Android Studio 2.2 Beta 2
HelloWorld > app > src > main > java > com > example > android > helloworld > MainActivity
Android
app
manifests
java
com.example.android.helloworld
MainActivity
com.example.android.helloworld (andr
com.example.android.helloworld (test)
res
drawable
layout
activity_main.xml
mipmap
MainActivity.java
Android Emulator - Nexus_5_API_23:5554
Hello World
Hello World!
Run: AVD: Nexus_5_API_23 app
4: Run
TODO
6: Android Monitor
0: Messages
Terminal
1
2

```java
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);
    }
}
```

```
09/09 16:26:31: Launching app
$ adb push /Users/tbove/AndroidStudioProjects/HelloWorld/app/build/outputs/apk/app-debug.apk /data/local/tmp/com.example.an
$ adb shell pm install -r "/data/local/tmp/com.example.android.helloworld"
    pkg: /data/local/tmp/com.example.android.helloworld
Success


$ adb shell am start -n "com.example.android.helloworld/com.example.android.helloworld.MainActivity" -a android.intent.acti
Connected to process 2840 on device Nexus_5_API_23 [emulator-5554]
I/art: Not late-enabling -Xcheck:jni (already on)
W/System: ClassLoader referenced unknown path: /data/app/com.example.android.helloworld-1/lib/x86_64
W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphics.drawable.VectorDrawableCo
D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true

                  [ 09-09 16:26:55.830  2840: 2840 D/         ]
                  HostConnection::get() New Host Connection established 0x7fd339eff300, tid 2840


                  [ 09-09 16:26:55.879  2840: 2856 D/         ]
                  HostConnection::get() New Host Connection established 0x7fd339eff6c0, tid 2856
I/OpenGLRenderer: Initialized EGL, version 1.4
```

Gradle build finished in 13s 291ms (2 minutes ago)

1. **Emulator**
2. **Run Pane**

THANK YOU