What is Android

Before learning all topics of android, it is required to know what is android.

Android is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

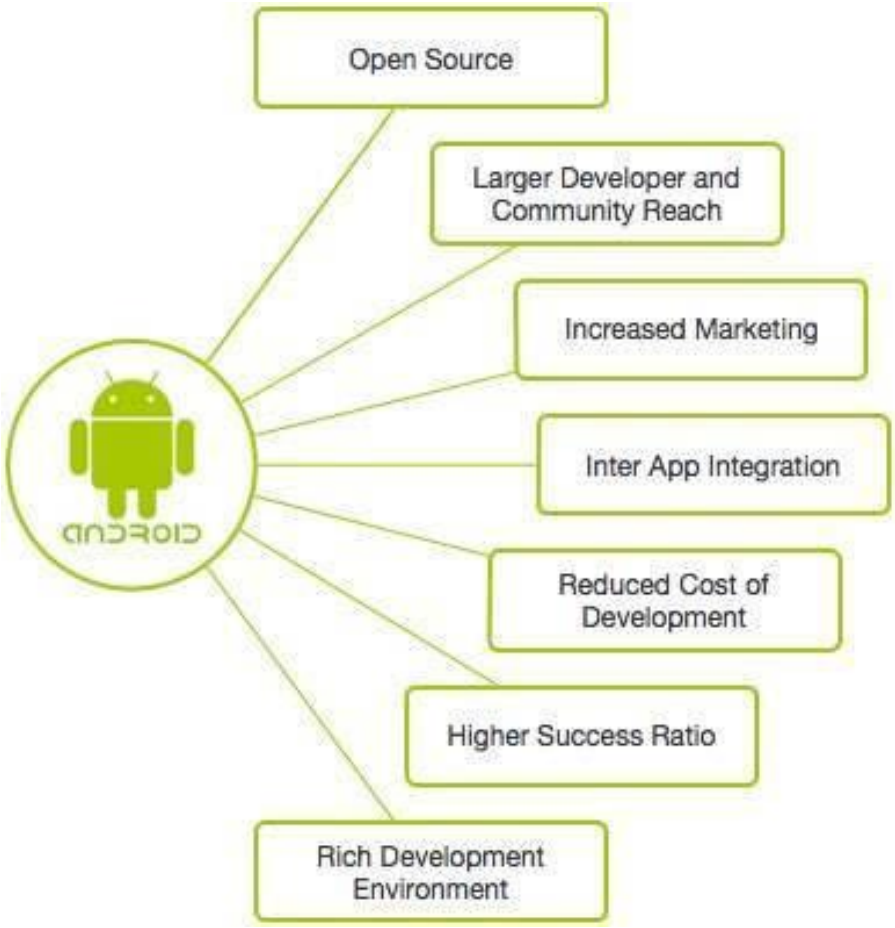
There are many code names of android such as Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc which is covered in next page.

What is Open Handset Alliance (OHA)

It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.

It was established on 5th November, 2007, led by Google. It is committed to advance open standards, provide services and deploy handsets using the Android Plateform.

Why Android ?



Features of Android

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below −

Sr.No.	Feature & Description
1	**Beautiful UI** Android OS basic screen provides a beautiful and intuitive user interface.
2	**Connectivity** GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
3	**Storage** SQLite, a lightweight relational database, is used for data storage purposes.

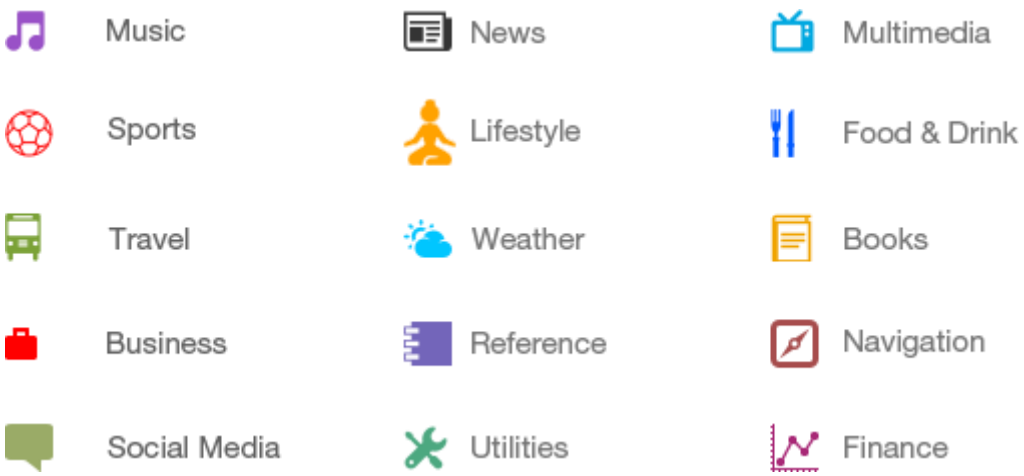
4	**Media support** H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
5	**Messaging** SMS and MMS
6	**Web browser** Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
7	**Multi-touch** Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.
8	**Multi-tasking** User can jump from one task to another and same time various application can run simultaneously.
9	**Resizable widgets** Widgets are resizable, so users can expand them to show more content or shrink them to save space.
10	**Multi-Language** Supports single direction and bi-directional text.
11	**GCM** Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
12	**Wi-Fi Direct** A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.

13	**Android Beam** A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together.

Categories of Android applications

There are many android applications in the market. The top categories are:

- Entertainment
- Tools
- Communication
- Productivity
- Personalization
- Music and Audio
- Social
- Media and Video
- Travel and Local etc.



History of Android

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as **Aestro**, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**, **Gingerbread**, **Honeycomb**, **Ice Cream Sandwitch**, **Jelly Bean**, **KitKat** and **Lollipop**. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**.

4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

5) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

6) In 2007, Google announces the development of android OS.

7) In 2008, HTC launched the first android mobile.

Android Versions, Codename and API

Let's see the android versions, codenames and API Level provided by Google.

Version	*Code name*	*API Level*
1.5	Cupcake	3
1.6	Donut	4
2.1	Eclair	7
2.2	Froyo	8
2.3	Gingerbread	9 and 10
3.1 and 3.3	Honeycomb	12 and 13
4.0	Ice Cream Sandwitch	15
4.1, 4.2 and 4.3	Jelly Bean	16, 17 and 18
4.4	KitKat	19
5.0	Lollipop	21
6.0	Marshmallow	23
7.0	Nougat	24-25
8.0	Oreo	26-27
9.0	Pie	28
10.0	Quince Tart	29
11.0	Red Velvet Cake	30
12.0	Snow Cone	31-32
*13.0**BETA***	Tiramisu	33

Version	SDK / API level	Version code	Codename	Cumulative usage [1]	Year
Android 13 BETA	Level 33	T	Tiramisu [2]	*No data*	*TBA*
Android 12	Level 32 Android 12L	S_V2	Snow Cone [2]		
	Level 31 Android 12	S		14.6%	2021
	▪ `targetSdk` will need to be 31+ for new apps by August 2022 and updates by November 2022. [3] ▪ `targetSdk` will need to be 31+ for *all existing apps* by November 2023. [4]				
Android 11	Level 30	R	Red Velvet Cake [2]	47.7%	2020
	▪ `targetSdk` must be 30+ for new apps and app updates. [3] ▪ `targetSdk` will need to be 30+ for *all existing apps* by November 2022. [4]				
Android 10	Level 29	Q	Quince Tart [2]	70.3%	2019
Android 9	Level 28	P	Pie	81.6%	2018
	▪ `targetSdk` must be 28+ for new Wear OS apps and Wear OS app updates.				
Android 8	Level 27 Android 8.1	O_MR1	Oreo	87.5%	2017
	Level 26 Android 8.0	O		90.2%	
Android 7	Level 25 Android 7.1	N_MR1	Nougat	91.8%	2016

	Level 24 **Android 7.0**	`N`		94.6%	
Android 6	Level 23	`M`	Marshmallow	97.1%	2015
Android 5	Level 22 **Android 5.1**	`LOLLIPOP_MR1`	Lollipop	98.7%	2015
	Level 21 **Android 5.0**	`LOLLIPOP`, `L`		*No data*	2014
	▪ Jetpack Compose requires a **`minSdk`** of 21 or higher.				
Android 4	Level 20 **Android 4.4W** [5]	`KITKAT_WATCH`	KitKat		
	Level 19 **Android 4.4**	`KITKAT`			2013
	▪ Google Play services do not support Android versions below API level 19.				
	Level 18 **Android 4.3**	`JELLYBEAN_MR2`	Jelly Bean		
	Level 17 **Android 4.2**	`JELLYBEAN_MR1`			2012
	Level 16 **Android 4.1**	`JELLYBEAN`			
	Level 15 **Android 4.0.3 – 4.0.4**	`ICE_CREAM_SANDWICH_MR1`	Ice Cream Sandwich		2011
	Level 14 **Android 4.0.1 – 4.0.2**	`ICE_CREAM_SANDWICH`			
	▪ Jetpack/AndroidX libraries require a **`minSdk`** of 14 or higher.				
Android 3	Level 13 **Android 3.2**	`HONEYCOMB_MR2`	Honeycomb		
	Level 12 **Android 3.1**	`HONEYCOMB_MR1`			

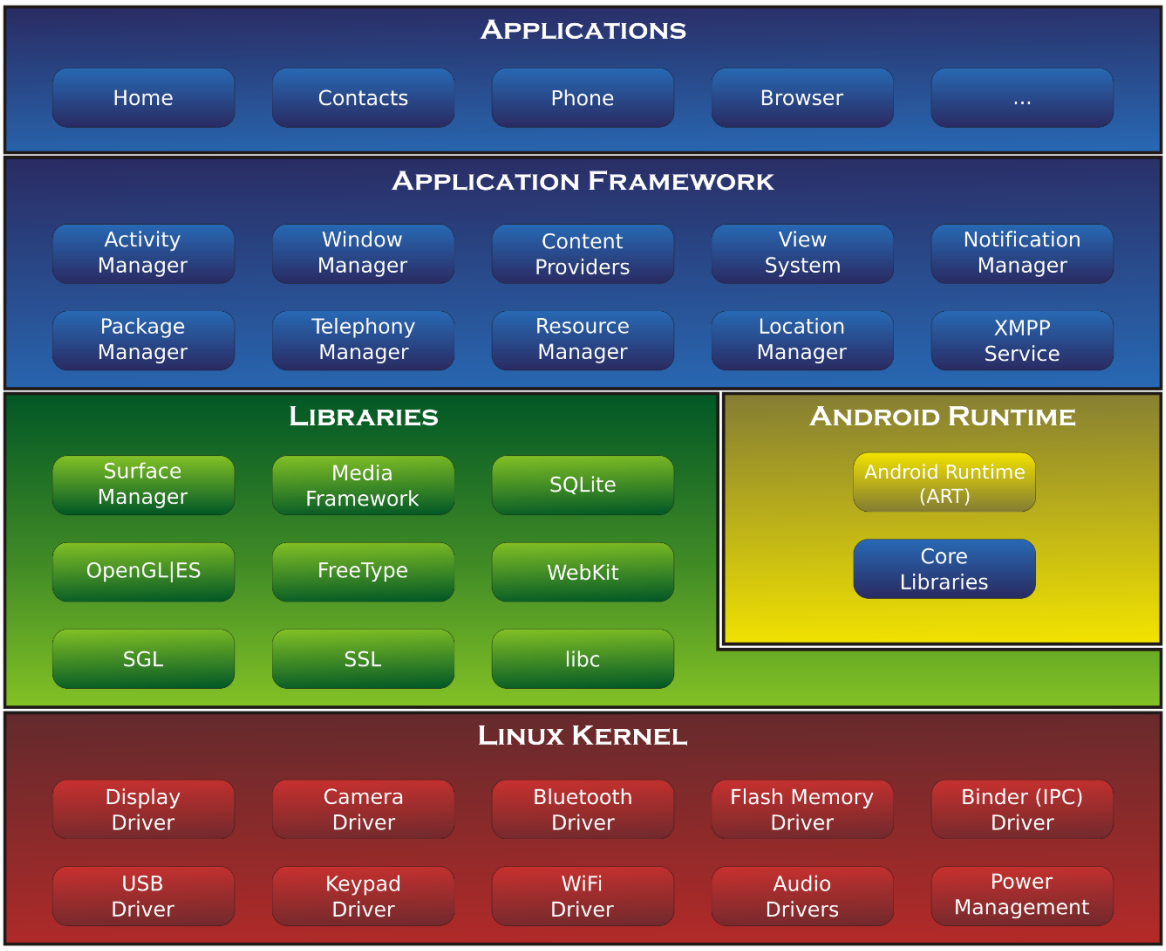
	Level 11 Android 3.0	HONEYCOMB			
Android 2	Level 10 Android 2.3.3 – 2.3.7	GINGERBREAD_MR1	Gingerbread		
	Level 9 Android 2.3.0 – 2.3.2	GINGERBREAD			2010
	Level 8 Android 2.2	FROYO	Froyo		
	Level 7 Android 2.1	ECLAIR_MR1	Eclair		
	Level 6 Android 2.0.1	ECLAIR_0_1			2009
	Level 5 Android 2.0	ECLAIR			
Android 1	Level 4 Android 1.6	DONUT	Donut		
	Level 3 Android 1.5	CUPCAKE	Cupcake		
	Level 2 Android 1.1	BASE_1_1	Petit Four		
	Level 1 Android 1.0	BASE	*None*		2008

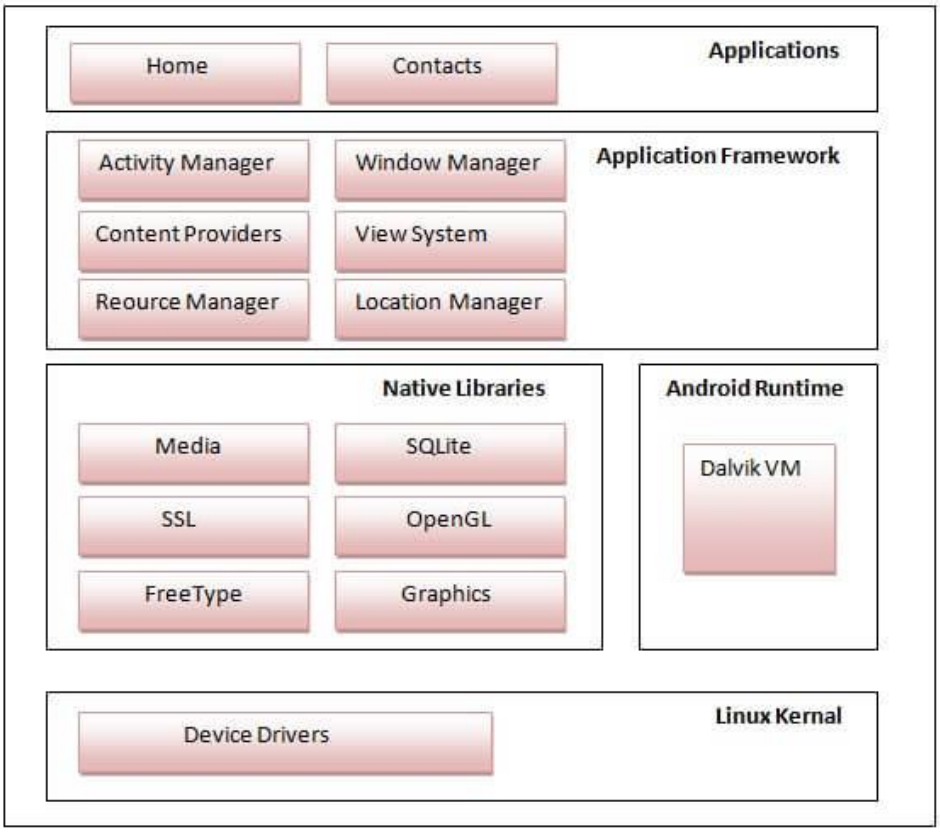
Android Architecture

android architecture or **Android software stack** is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.


APPLICATIONS
Home
Contacts
Phone
Browser
...
APPLICATION FRAMEWORK
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource Manager
Location Manager
XMPP Service
LIBRARIES
Surface Manager
Media Framework
SQLite
OpenGL|ES
FreeType
WebKit
SGL
SSL
libc
ANDROID RUNTIME
Android Runtime (ART)
Core Libraries
LINUX KERNEL
Display Driver
Camera Driver
Bluetooth Driver
Flash Memory Driver
Binder (IPC) Driver
USB Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management


Home
Contacts
Applications
Activity Manager
Window Manager
Application Framework
Content Providers
View System
Reource Manager
Location Manager
Native Libraries
Media
SQLite
SSL
OpenGL
FreeType
Graphics
Android Runtime
Dalvik VM
Device Drivers
Linux Kernal

1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of linux kernel, their are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

Android Core Building Blocks

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.

View

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent

Intent is used to invoke components. It is mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

For example, you may write the following code to view the webpage.

```
1. Intent intent=new Intent(Intent.ACTION_VIEW);
2. intent.setData(Uri.parse("http://www.nm-aist.ac.tz"));
3. startActivity(intent);
```

Service

Service is a background process that can run for a long time.

There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider

Content Providers are used to share data between the applications.

Fragment

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml

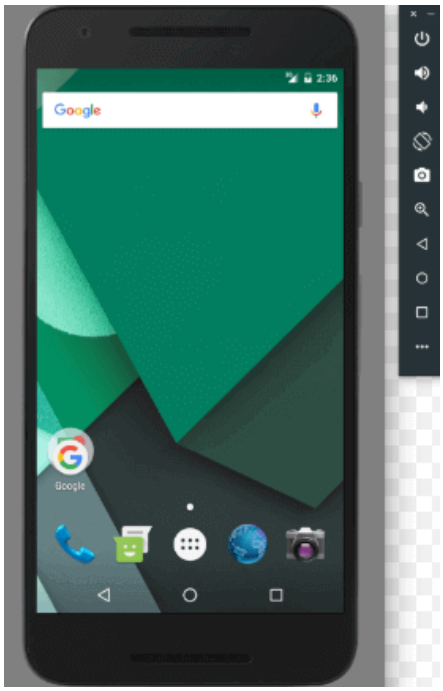
It contains informations about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

Android Emulator

The **Android emulator** is an **Android Virtual Device (AVD),** which represents a specific Android device. We can use the Android emulator as a target device to execute and test our Android application on our PC. The Android emulator provides almost all the functionality of a real device. We can get the incoming phone calls and text messages. It also gives the location of the device and simulates different network speeds. Android emulator simulates rotation and other hardware sensors. It accesses the Google Play store, and much more



Testing Android applications on emulator are sometimes faster and easier than doing on a real device. For example, we can transfer data faster to the emulator than to a real device connected through USB.

The Android emulator comes with predefined configurations for several Android phones, Wear OS, tablet, Android TV devices.

Requirement and recommendations

The Android emulator takes additional requirements beyond the basic system requirement for Android Studio. These requirements are given below:

- SDK Tools 26.1.1 or higher
- 64-bit processor
- Windows: CPU with UG (unrestricted guest) support
- HAXM 6.2.1 or later (recommended HAXM 7.2.0 or later)

Install the emulator

The Android emulator is installed while installing the Android Studio. However some components of emulator may or may not be installed while installing Android Studio. To install the emulator component, select the **Android Emulator** component in the **SDK Tools** tab of the **SDK Manager.**

Run an Android app on the Emulator

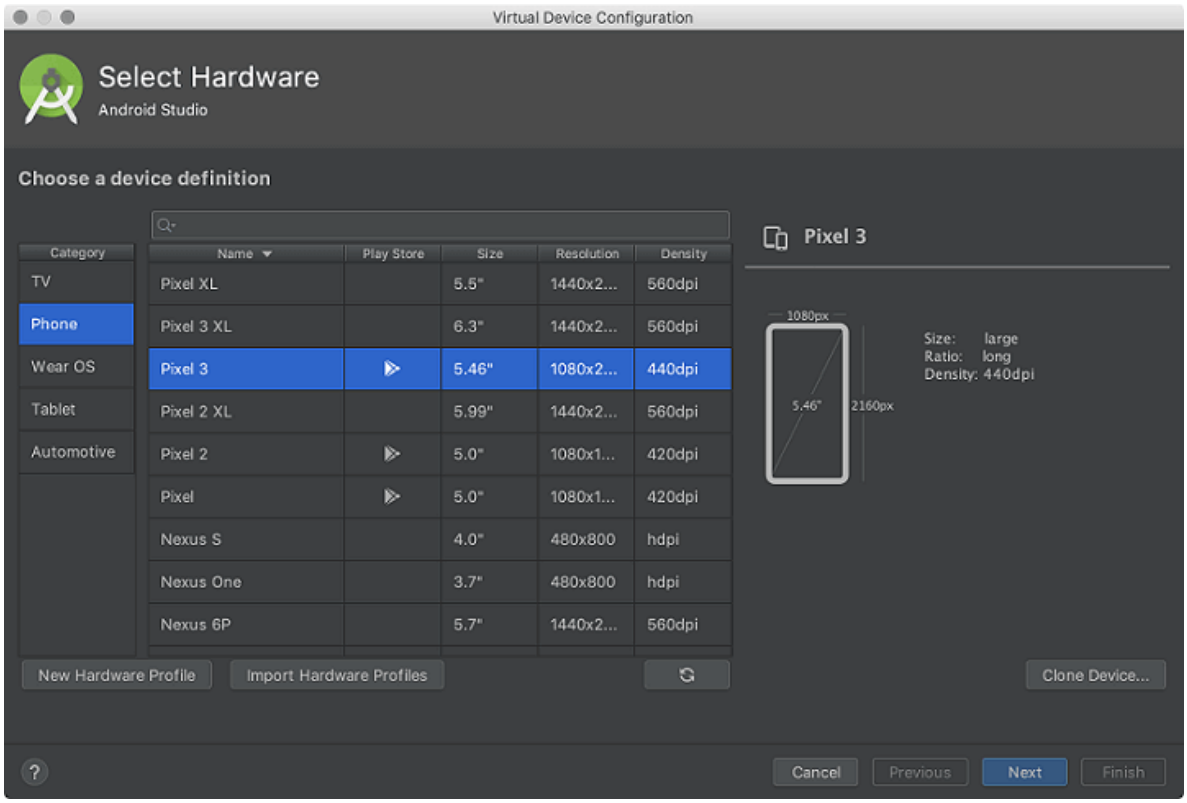
We can run an Android app form the Android Studio project, or we can run an app which is installed on the Android Emulator as we run any app on a device.

To start the Android Emulator and run an application in our project:

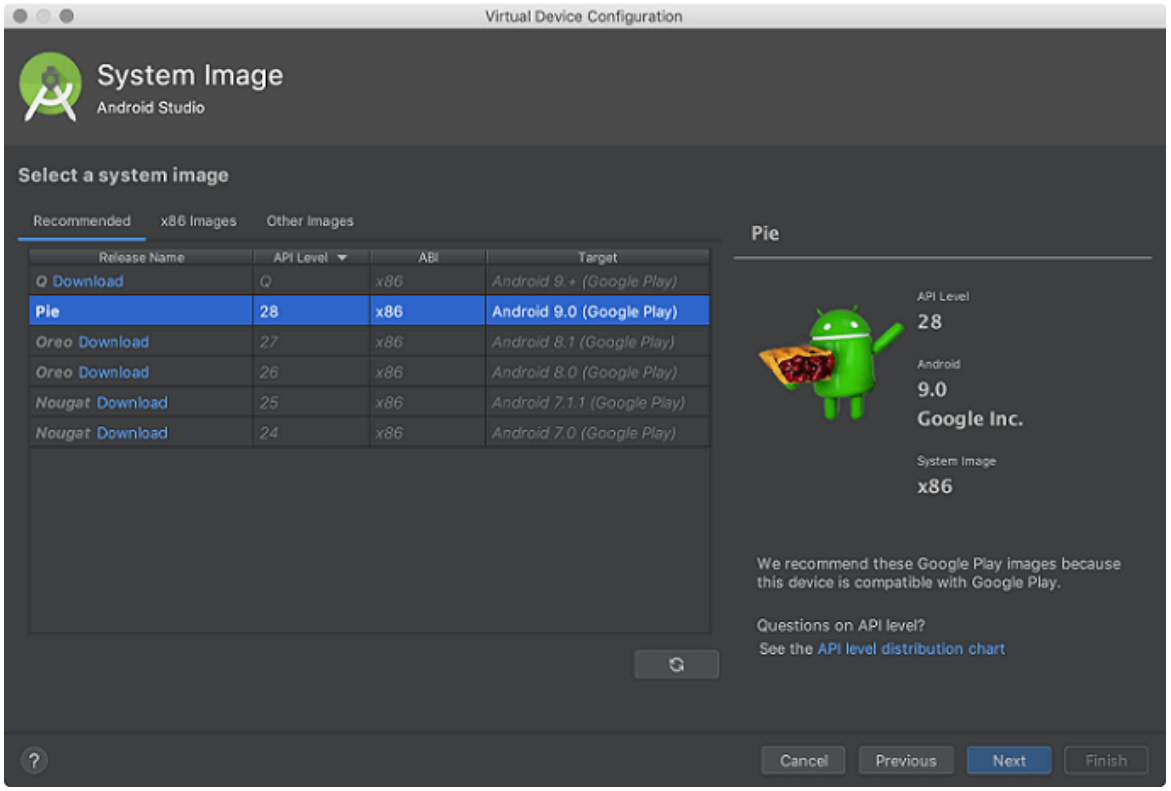
1. In **Android Studio,** we need to create an Android Virtual Device (AVD) that the emulator can use to install and run your app. To create a new AVD:-

1.1 Open the AVD Manager by clicking **Tools > AVD Manager.**

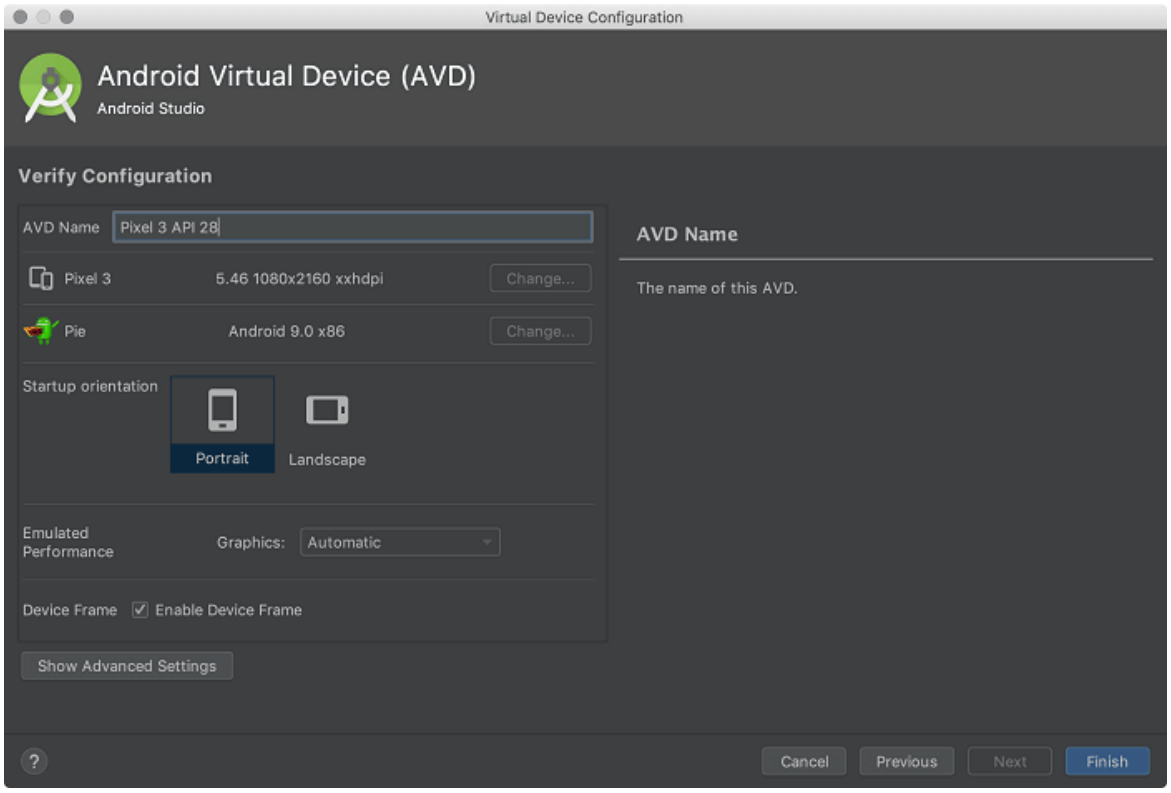
1.2 Click on Create **Virtual** Device, at the bottom of the AVD Manager dialog. Then **Select Hardware** page appears.



1.3 Select a hardware profile and then click **Next.** If we don?t see the hardware profile we want, then we can create or import a hardware profile. The **System Image** page appears.

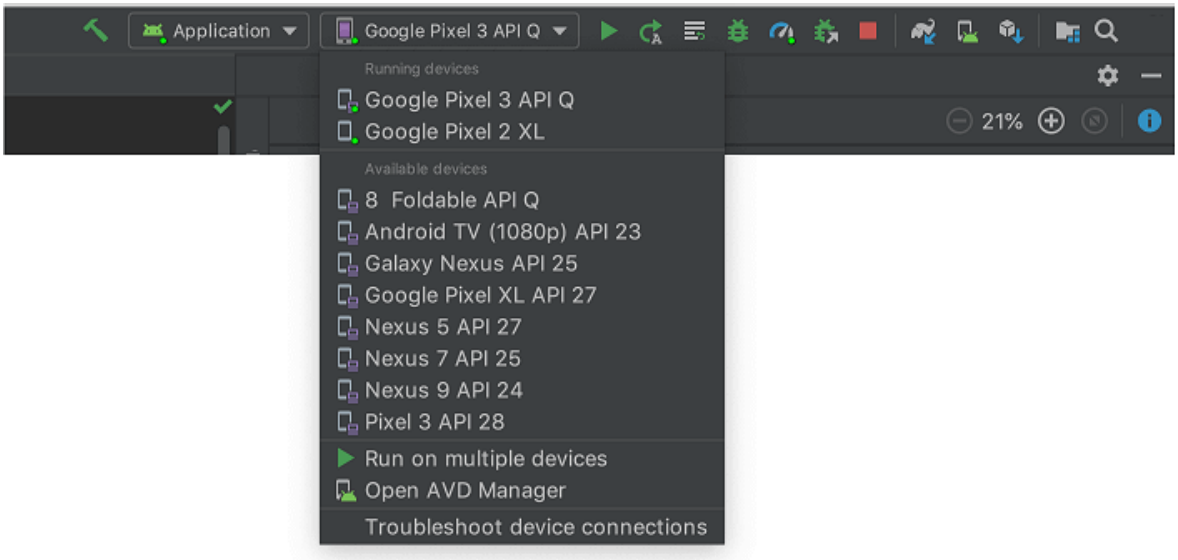


1.4 Select the system image for the particular API level and click **Next.** This leads to open a **Verify Configuration** page.



1.5 Change AVD properties if needed, and then click **Finish.**

2. In the toolbar, choose the AVD, which we want to run our app from the target device from the drop-down menu.



3. Click **Run.**

Launch the Emulator without first running an app

To start the emulator:

1. Open the AVD Manager.
2. Double-click an AVD, or click **Run**

While the emulator is running, we can run the Android Studio project and select the emulator as the target device. We can also drag an APKs file to install on an emulator, and then run them.

Start the emulator from the command line

The Android SDK includes the Android device emulator. Android emulator lets you develop and test out the application without using a physical device.

Starting the emulator

Using the **emulator** command, we will start an emulator. It is an alternative to run our project or start through the AVD Manager.

Here is the basic command-line syntax for starting a virtual device:

1. $ emulator -avd avd_name [{-option [value]} ...]

or

1. $ emulator @avd_name [{-option [value]} ...]

For example, if we execute the emulator from Android Studio on a Mac, the default command line will be similar as follows:

1. $ /Users/user_name/Library/Android/sdk/emulator/emulator -avd Nexus_5X_API_23 -netdelay none -netspeed full

To display the list of AVD names, enter the following command:

1. $ emulator -list-avds

Run and stop an emulator, and clear data

From the Virtual Device page, we can perform the following operation on emulator:

- To run an Android emulator that uses an AVD, double-click the AVD, or click **Launch**
- To stop the running emulator, right-click and select **Stop,** or click Menu ▼ and select Stop.
- If we want to clear the data from an emulator and return it to the initial state when it was first defined, then right-click an AVD and select **Wipe Data.** Or click menu ▼ and select **Wipe Data.**