**Introduction to Android :**

Android is a [mobile operating system](https://en.wikipedia.org/wiki/Mobile_operating_system) based on a modified version of the [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) and other [open source](https://en.wikipedia.org/wiki/Open-source_software) software, designed primarily for [touch screen](https://en.wikipedia.org/wiki/Touchscreen) mobile devices such as [smart phones](https://en.wikipedia.org/wiki/Smartphone) and [tablets](https://en.wikipedia.org/wiki/Tablet_computer). Android is developed by a consortium of developers known as the [Open Handset Alliance](https://en.wikipedia.org/wiki/Open_Handset_Alliance), with the main contributor and commercial marketer being [Google](https://en.wikipedia.org/wiki/Google).

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the [first commercial Android device](https://en.wikipedia.org/wiki/HTC_Dream) launched in September 2008. The current stable version is [Android 10](https://en.wikipedia.org/wiki/Android_10), released on September 3, 2019. The core Android source code is known as Android Open Source Project (AOSP), which is primarily licensed under the [Apache License](https://en.wikipedia.org/wiki/Apache_License). This has allowed variants of Android to be developed on a range of other electronics, such as [game consoles](https://en.wikipedia.org/wiki/Video_game_console), [digital cameras](https://en.wikipedia.org/wiki/Digital_camera), [PCs](https://en.wikipedia.org/wiki/Personal_computer) and others, each with a specialized user interface. Some well known derivatives include [Android TV](https://en.wikipedia.org/wiki/Android_TV) for televisions and [Wear OS](https://en.wikipedia.org/wiki/Wear_OS) for wearable’s, both developed by Google.

Android's source code has been used as the basis of different ecosystems, most notably that of Google which is associated with a suite of [proprietary software](https://en.wikipedia.org/wiki/Proprietary_software) called [Google Mobile Services](https://en.wikipedia.org/wiki/Google_Mobile_Services) (GMS), that frequently comes pre-installed on said devices. This includes core apps such as [Gmail](https://en.wikipedia.org/wiki/Gmail), the [digital distribution](https://en.wikipedia.org/wiki/Digital_distribution) platform [Google Play](https://en.wikipedia.org/wiki/Google_Play) and associated [Google Play Services](https://en.wikipedia.org/wiki/Google_Play_Services) development platform, and usually apps such as the [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome) web browser. These apps are licensed by manufacturers of Android devices certified under standards imposed by Google. Other competing Android ecosystems include [Amazon.com](https://en.wikipedia.org/wiki/Amazon.com)'s [Fire OS](https://en.wikipedia.org/wiki/Fire_OS), or [LineageOS](https://en.wikipedia.org/wiki/LineageOS). Software distribution is generally offered through proprietary [application stores](https://en.wikipedia.org/wiki/Application_store) like [Google Play Store](https://en.wikipedia.org/wiki/Google_Play_Store) or [Samsung Galaxy Store](https://en.wikipedia.org/wiki/Samsung_Galaxy_Store), or open source platforms like [Aptoide](https://en.wikipedia.org/wiki/Aptoide) or [F-Droid](https://en.wikipedia.org/wiki/F-Droid), which use software packages in the [APK](https://en.wikipedia.org/wiki/Android_application_package) format.

The OS first OS version was introduced in 2007 with many of its versions named in Alphabetical order ranging from A-N and upcoming is Q.

**HERE IS DETAILED ABOUT ANDROID VERSIONS :**

**Alpha –** In this( Android 1.0) was the first versions of Android operating System by Google. It has basic functionality with a simple browser and other Google apps like Gmail, Maps and YouTube.

**Beta –** Later on with Android 1.1 few more functionality added, the API changes from Level 1 in Android 1.0 to Level 2. It supports attachment with MMS.

**Cupcake –** Cupcake was Android second version with new features as well as the Android framework API updated. It was Android 1.5 with on Screen Keyboard , Bluetooth and Updated UI for applications.

**Donut –** It was Android 1.6 nicknamed as DONUT. It added support for CDMA , additional screen sizes, talk to speech engine and battery indicator.

**Eclair –** Android 2.0-2.1 as like other versions this also come up with a nickname as ECLAIR and lot more functions & features. It come up with Bluetooth 2.1 , live wallpaper, HTML 5 support, ability to search sms & mms, flash support, digital zoom and more camera features

**Froyo –** Android version 2.2-2.2.3 introduced with USB tethering & WiFi hotspot functionality and apps can now be installed on memory card. Support Adobe flash, increased speed and performance of applications with new features.

**Gingerbread –** Gingerbread (Android 2.3-2.3.7) introduced with updated User Interface which provide more ease to use. Features are like sensors, multiple cameras(Front & back), virtual keyboard, better text suggestion, voice input capability and press hold copy paste capability.

**Honeycomb –** This Android platform Honeycomb was designed for large screens like tablets so interface elements like virtual keyboard optimized for bigger screen. Home screen is optimized, tabs are introduced in browser with additional incognito mode and video chat & Gtalk is supported.

**Ice Cream Sandwich –**Ice Cream sandwich come in 2011 bringing all new look. It gives more ease to user like user can quickly swipe to close the apps, new gallery layout and built in photo editor.

**Jelly Bean –** Google made Operating System more responsive with Jelly Bean and introduces file sharing with Android Beam. Restricted profile, Dial Pad complete, supported other languages like Hindi, changed camera UI.

**KitKat –** Kitkat come up in 2013 with API Level 19. It has wireless printing capability, new dailer id, chrome webview and screen recording.

**Lollipop –** Android version 5.0-5.1.1 come up with improved RAM and battery management. Further restyling through Material design, no interrupts feature, unlock phone through Bluetooth trusted devices, print previews and smart lock feature.

**Marshmallow –** Marshmallow was released in year 2015, come up with smarter battery and doze mode (it prevents certain task from running if the phone being setting idle), Now On Tap, better privacy settings, easier to upgrade phone, Fingerprint sensor and built in visual voice mail.

**Nougat –** Android Nougat was made official in 2016 with updated emoji, 72 newly added, multi window view (switch between apps with double tap), smarter battery with data saver mode, more secured and high quality virtual reality with new dimensions.

**ANDROID PLATFORM VERSIONS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Codename** | **API** | **Distribution** |
| **2.2** | **Froyo** | **8** | **0.1%** |
| **2.3.3 – 2.3.7** | **Gingerbread** | **10** | **1.2%** |
| **4.0.3 – 4.0.4** | **Ice Cream Sandwich** | **15** | **1.2%** |
| **4.1.X** | **Jelly Bean** | **16** | **4.5%** |
| **4.2.X** | **Jelly Bean** | **17** | **6.4%** |
| **4.3** | **Jelly Bean** | **18** | **1.9%** |
| **4.4** | **Kitkat** | **19** | **24.0%** |
| **5.0** | **Lollipop** | **21** | **10.8%** |
| **5.1** | **Lollipop** | **22** | **23.2%** |
| **6.0** | **Marshmallow** | **23** | **26.3%** |
| **7.0** | **Nougat** | **24** | **0.4%** |

**Applications and Features:**

You will find all [the android applications](https://www.edgefxkits.com/home-automation-by-android-application-based-remote-control?utm_source=elprocus.com) at the top layer and you will write your application and install on this layer. Example of such applications are contacts, books, browsers, services etc. Each application perform a different role in the over all applications.

**Features:**

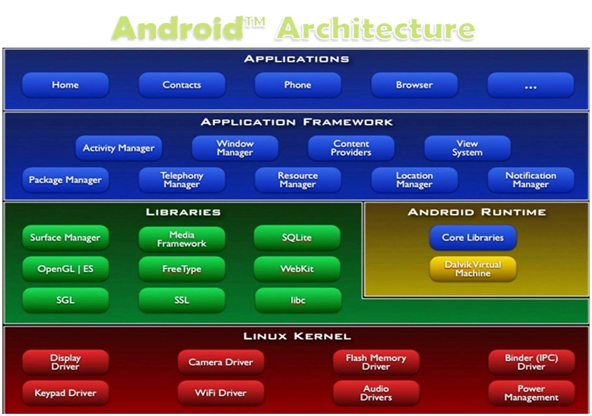
* Head set layout
* Storage
* Connectivity: GSM/EDGE, IDEN, CDMA, Bluetooth, WI-FI, EDGE,3G,NFC, LTE,GPS.
* Messaging: SMS, MMS, C2DM (could to device messaging), GCM (Google could messaging)
* Multilanguage support
* Multi touch
* Video calling
* Screen capture
* External storage
* Streaming media support
* Optimized graphics

### Android Architecture:

The android is a operating system and is a stack of software components which is divided into five sections and four main layers that is

* Linux kernel
* Libraries
* Android runtime

### Application frame work:

[](https://www.elprocus.com/wp-content/uploads/2013/10/Android-Architecture.png)

### Linux kernel:

The android uses the powerful Linux kernel and it supports wide range of hardware drivers. The kernel is the heart of the operating system that manages input and output requests from software. This provides basic system functionalities like process management, memory management, device management like camera, keypad, display etc the kernel handles all the things. The Linux is really good at networking and it is not necessary to interface it to the peripheral hardware. The kernel itself does not interact directly with the user but rather interacts with the shell and other programs as well as with the hard ware devices on the system.

### Libraries:

The on top of a Linux kennel there is a set of libraries including open source web browser such as webkit, library libc. These libraries are used to play and record audio and video. The SQLite is a data base which is useful for storage and sharing of application data. The SSL libraries are responsible for internet security etc.

### Android Runtime:

The android runtime provides a key component called Dalvik Virtual Machine which is a kind of java virtual machine. It is specially designed and optimized for android. The Dalvik VM is the process virtual machine in the android operating system. It is a software that runs apps on android devices.

The Dalvik VM makes use of Linux core features like memory management and multithreading which is in a java language. The Dalvik VM enables every android application to run it own process. The Dalvik VM  executes the files in the .dex format.

### Application frame work:

The application frame work layer provides many higher level services to applications such as windows manager, view system, package manager, resource manager etc. The application developers are allowed to make use of these services in their application.

**Creating Android Virtual Devices (AVDs)**

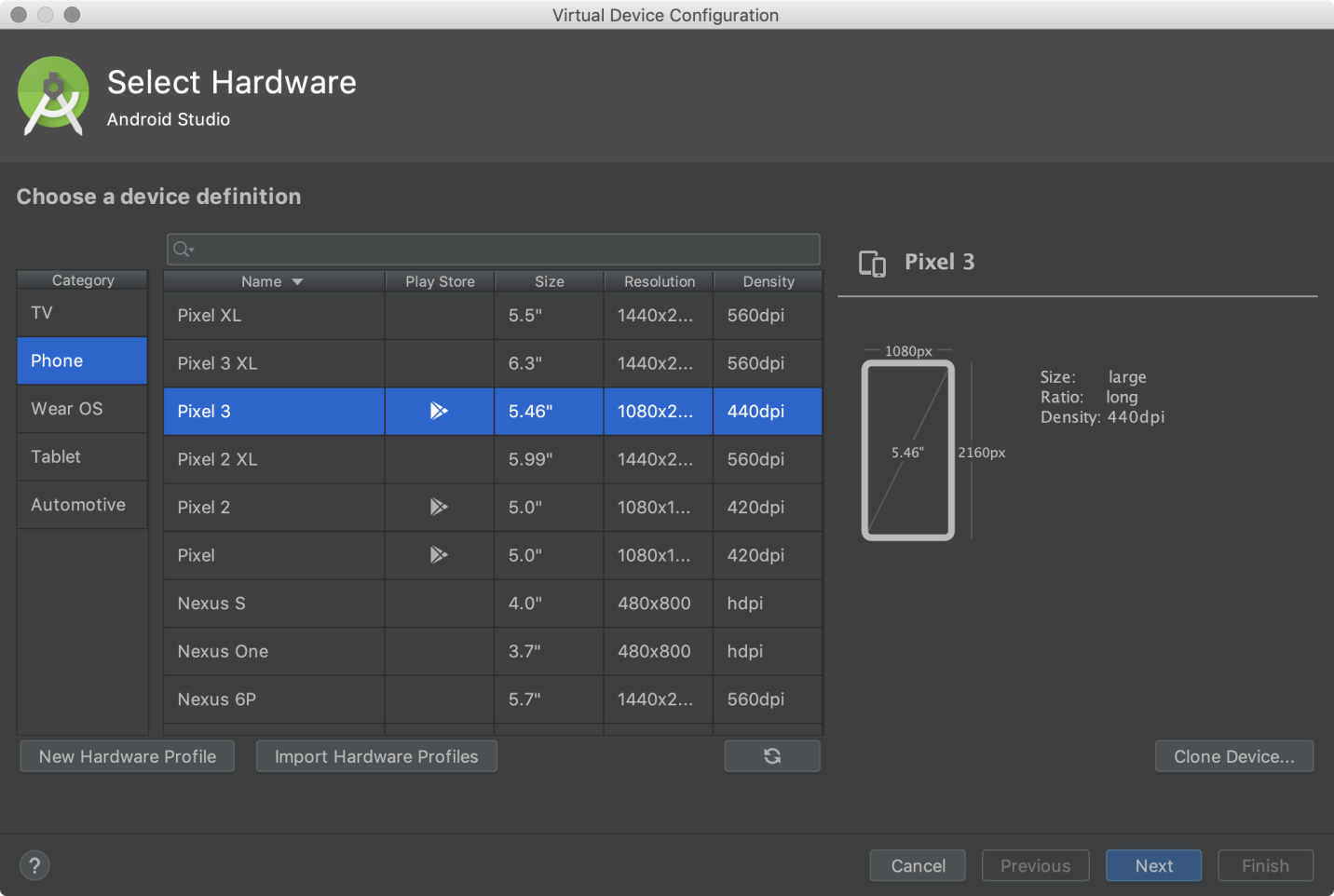
**To create a new AVD:**

1. Open the **AVD** Manager by clicking Tools > **AVD** Manager.
2. Click **Create Virtual Device**, at the bottom of the **AVD** Manager dialog. ...
3. Select a hardware profile, and then click Next.
4. Select the system image for a particular API level, and then click Next.
5. Change **AVD** properties as needed, and then click Finish.

To create a new AVD:

1. Open the AVD Manager by clicking **Tools > AVD Manager**. 
2. Click **Create Virtual Device**, at the bottom of the AVD Manager dialog.

The **Select Hardware** page appears.

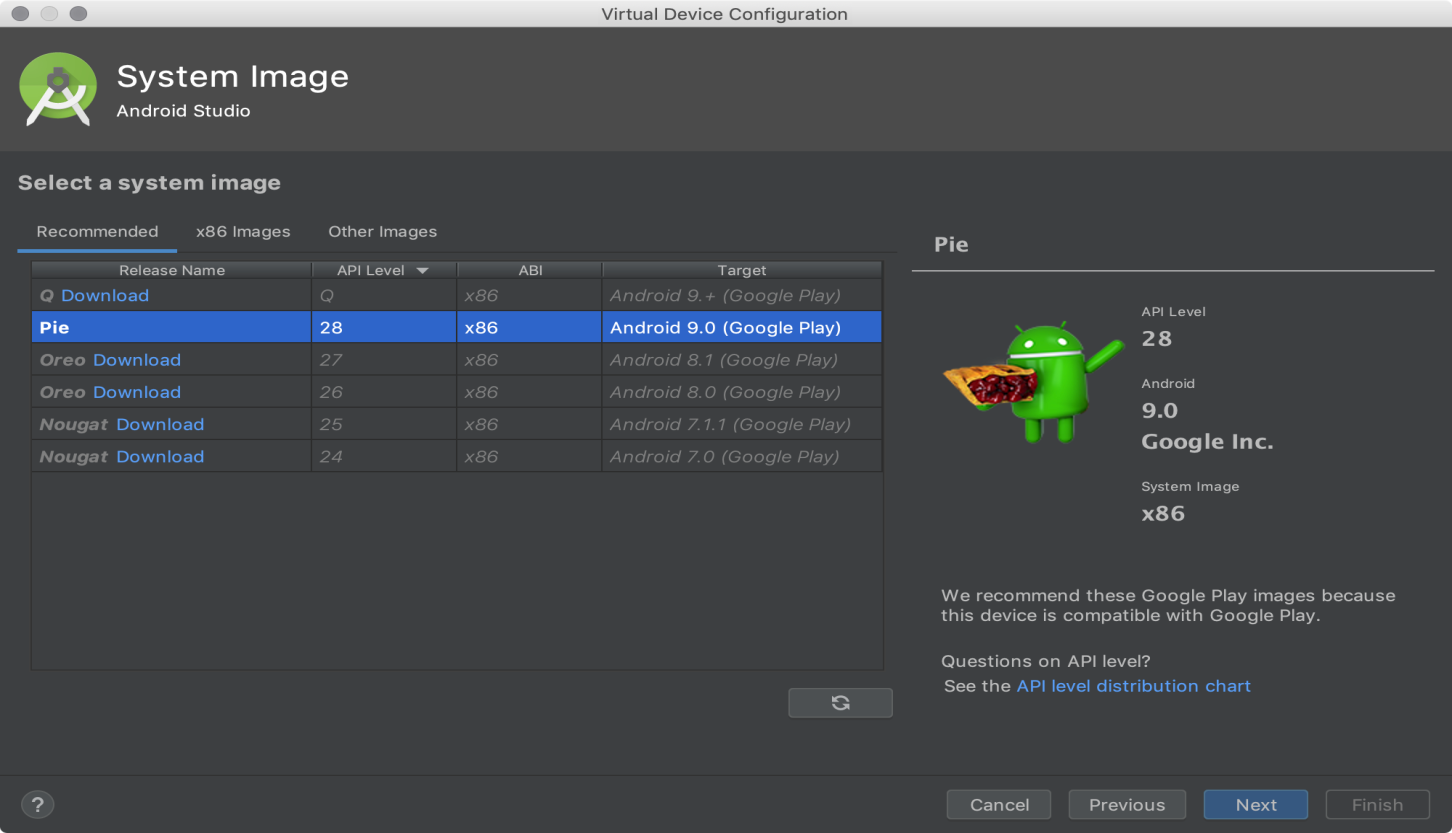


Notice that only some hardware profiles are indicated to include **Play Store**. This indicates that these profiles are fully [CTS](https://source.android.com/compatibility/cts/) compliant and may use system images that include the Play Store app.

1. Select a hardware profile, and then click **Next**.

If you don't see the hardware profile you want, you can [create](https://developer.android.com/studio/run/managing-avds#createhp) or [import](https://developer.android.com/studio/run/managing-avds#importexporthp) a hardware profile.

The **System Image** page appears.



1. Select the system image for a particular API level, and then click **Next**.

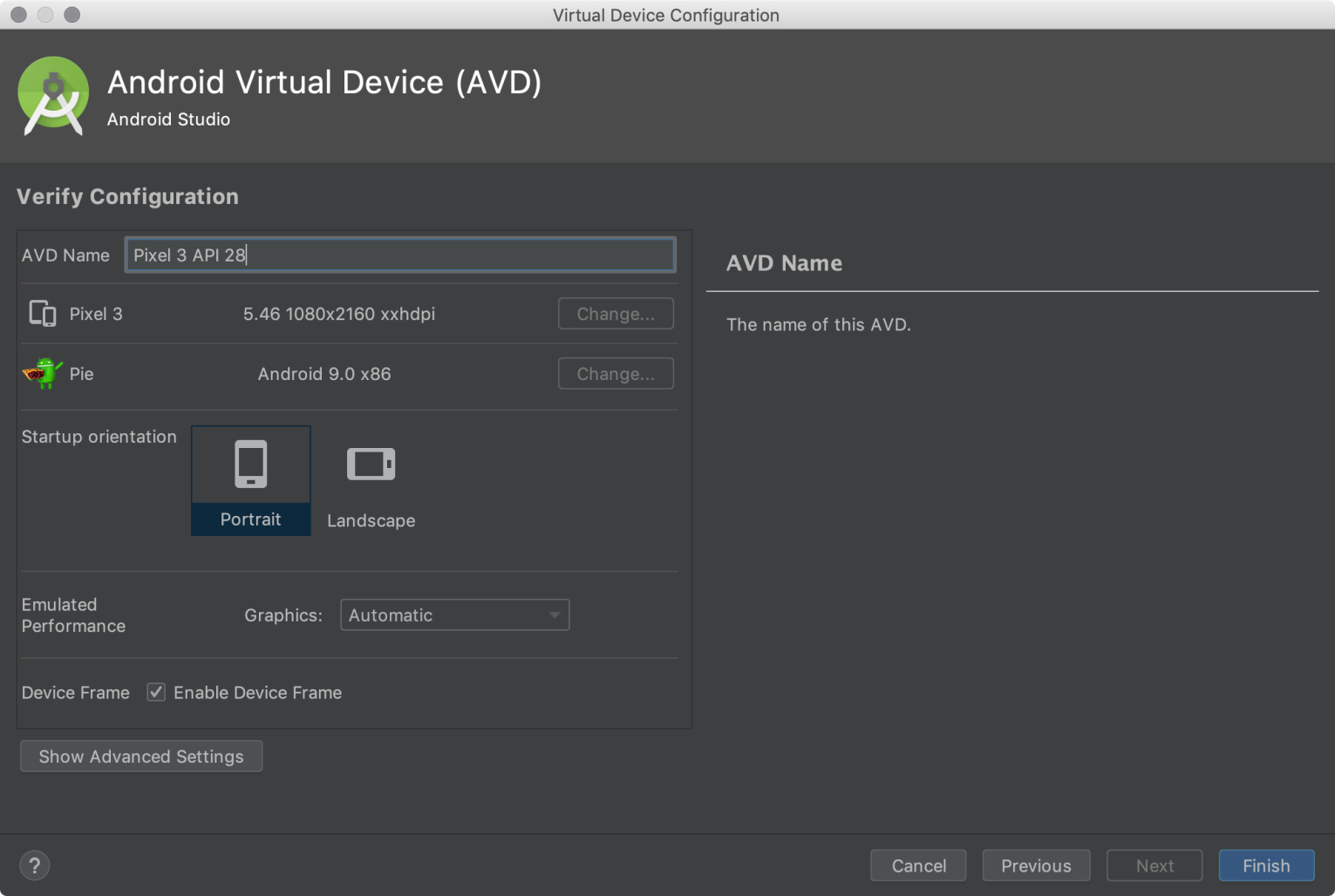
The **Recommended** tab lists recommended system images. The other tabs include a more complete list. The right pane describes the selected system image. x86 images run the fastest in the emulator.

If you see **Download** next to the system image, you need to click it to download the system image. You must be connected to the internet to download it.

The API level of the target device is important, because your app won't be able to run on a system image with an API level that's less than that required by your app, as specified in the [minSdkVersion](https://developer.android.com/guide/topics/manifest/uses-sdk-element) attribute of the app manifest file. For more information about the relationship between system API level and minSdkVersion, see [Versioning Your Apps](https://developer.android.com/studio/publish/versioning).

If your app declares a [<uses-library>](https://developer.android.com/guide/topics/manifest/uses-library-element) element in the manifest file, the app requires a system image in which that external library is present. If you want to run your app on an emulator, create an AVD that includes the required library. To do so, you might need to use an add-on component for the AVD platform; for example, the Google APIs add-on contains the Google Maps library.

The **Verify Configuration** page appears.



1. Change [AVD properties](https://developer.android.com/studio/run/managing-avds#avdproperties) as needed, and then click **Finish**.

**Types of Android Application**

**Three Types of Apps: Native, Hybrid, and Web**

**1.Native mobile apps:**

Native mobile apps are designed to be “native” to one platform, whether it’s Apple iOS, Google’s Android, or Windows Phone. The native platform can be advantageous because it tends to optimize the user experience. Because it was developed specifically for the platform, it can operate more quickly and intuitively.

**Advantages of native apps:**

1. Natives are very fast.  
2. Easily distributed in google apple app stores.  
3. More interactive and intuitive.  
4. Easily interact with any feature of the phone.

**Disadvantages of native apps:**

1. Built for a single platform  
2. Languages like swift and java used to build these types of apps are hard to learn.  
3. Expensive to develop.  
4. hard to maintain.

**2. Hybrid mobile apps:**

These apps can be installed on devices just like native apps, but they run through web browsers. All hybrid apps are developed through the HTML5 programming language. Though hybrid apps are not as fast or reliable as native apps, they have a greater capacity for streamlining the development process. Because you don’t have to build and maintain apps for separate platforms, your business can save on time and resources. It’s ideal for apps that primarily deliver content.

**Advantages of hybrid apps:**

1. Easy to build  
2. Much cheaper than a native app  
3. Single app for all platforms.  
4. No browser needed  
5. Can usually access device utilities using an API  
6. Faster to develop than native apps.

**Disadvantages of Hybrid apps:**

1. Slower than native apps  
2. more expensive than web apps  
3. Less interactive than native apps

**3. Web apps:**

Responsive websites switch to a different design when they are accessed from a mobile device. Adaptive web applications, on the other hand, scale to fit the different screen sizes of mobile devices. For these apps, the design doesn’t change. Web apps are built using the most popular programming languages, but they can’t use hardware on mobile devices or be sold in any app stores.

**Advantages of web apps:**

1. Reduced business cost.  
2. No installation needed.  
3. Better reach as it can be accessed from anywhere.  
4. Always up-to-date.

**Disadvantages of web apps:**

1. Dependent on internet speed.  
2. Interface not that sophisticated.  
3. Take a longer time to develop.  
4. Security risk.

**Anatomy of an Android Application.**

There are four building blocks to an Android application:

* Activity
* Intent Receiver
* Service
* Content Provider

Not every application needs to have all four, but your application will be written with some combination of these.

Once you have decided what components you need for your application, you should list them in a file called AndroidManifest.xml. This is an XML file where you declare the components of your application and what their capabilities and requirements are. We will discuss soon, what the AndroidManifest.xml is responsible for.

**Activity** :

Activities are the most common of the four Android building blocks. An activity is usually a single screen in your application. Each activity is implemented as a single class that extends the Activity base class. Your class will display a user interface composed of Views and respond to events. Most applications consist of multiple screens. For example, a text messaging application might have one screen that shows a list of contacts to send messages to, a second screen to write the message to the chosen contact, and other screens to review old messages or change settings. Each of these screens would be implemented as an activity.

When a new screen opens, the previous screen is paused and put onto a history stack. The user can navigate backward through previously opened screens in the history. Screens can also choose to be removed from the history stack when it would be inappropriate for them to remain. Android retains history stacks for each application launched from the home screen.

**Intent and Intent Filters :**

Android uses a special class called Intent to move from screen to screen. Intent describe what an application wants done. The two most important parts of the intent data structure are the action and the data to act upon. Typical values for action are MAIN (the front door of the application), VIEW, PICK, EDIT, etc. The data is expressed as a Uniform Resource Indicator (URI). For example, to view a website in the browser, you would create an Intent with the VIEW action and the data set to a Website-URI.

new Intent(android.content.Intent.VIEW\_ACTION, ContentURI.create("http://anddev.org"));

There is a related class called an IntentFilter. While an intent is effectively a request to do something, an intent filter is a description of what intents an activity (or intent receiver, see below) is capable of handling. Activities publish their IntentFilters in the AndroidManifest.xml file.

**Intent Receiver :**

You can use an IntentReceiver when you want code in your application to execute in reaction to an external event, for example, when the phone rings, or when the data network is available, or when it's midnight. Intent receivers do not display a UI, although they may display Notifications to alert the user if something interesting has happened. Intent receivers are also registered in AndroidManifest.xml, but you can also register them from code using Context.registerReceiver().

**Service :**

A Service is code that is long-lived and runs without a UI. A good example of this is a media player playing songs from a play list. In a media player application, there would probably be one or more activities that allow the user to choose songs and start playing them. However, the music playback itself should not be handled by an activity because the user will expect the music to keep playing even after navigating to a new screen. In this case, the media player activity could start a service using Context.startService() to run in the background to keep the music going. The system will then keep the music playback service running until it has finished. (You can learn more about the priority given to services in the system by reading Life Cycle of an Android Application.) Note that you can connect to a service (and start it if it's not already running) with the Context.bindService() method. When connected to a service, you can communicate with it through an interface exposed by the service. For the music service, this might allow you to pause, rewind, etc.

**Content Provider :**

Applications can store their data in files, a SQLite database, preferences or any other mechanism that makes sense. A content provider, however, is useful if you want your application's data to be shared with other applications. A content provider is a class that implements a standard set of methods to let other applications store and retrieve the type of data that is handled by that content provider.