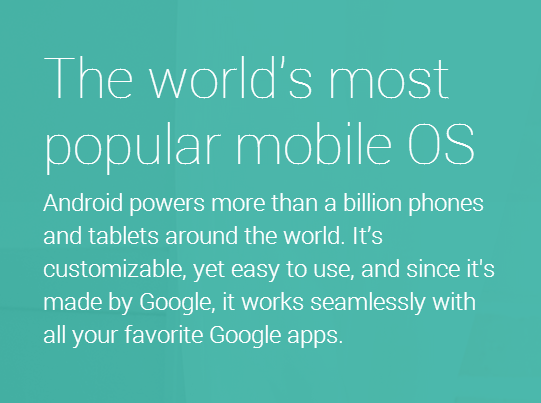
* Introduction to Android Platform
* Installations Required
* Understanding Android Virtual Device
* Connecting a real device
* Understanding Android SDK Manager
* Android ADB
* DDMS



What is Android?



Android versions identity

Each android release has two separate identities

1. Platform version – Identified by the decimal formatted values 2.1 , 4.1 , 4.4.1 etc.
2. API level – Important to developers (different naming system – simple Integer). It exposes the various SDK features. Ex. 1, 2…..18
3. Platform version and the API relationship – Each platform version supports a specific API level.

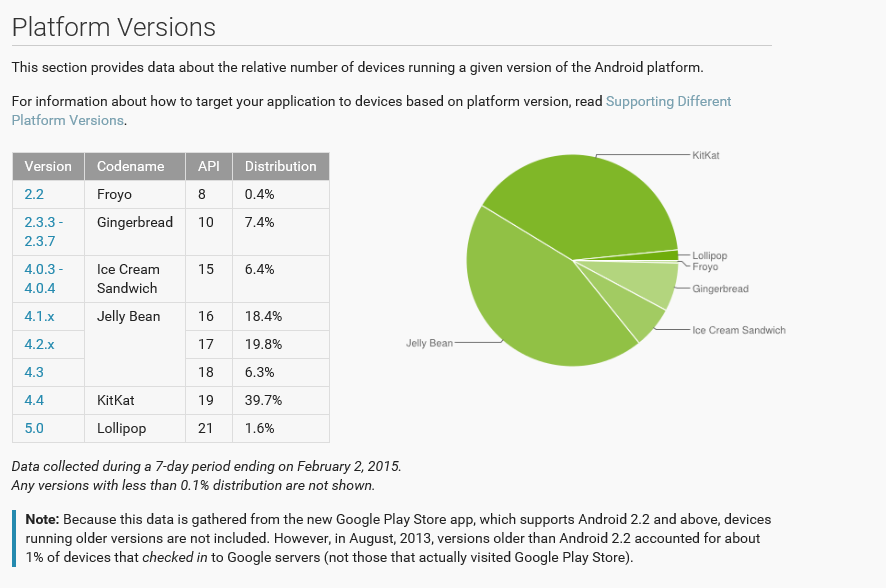


Documentation

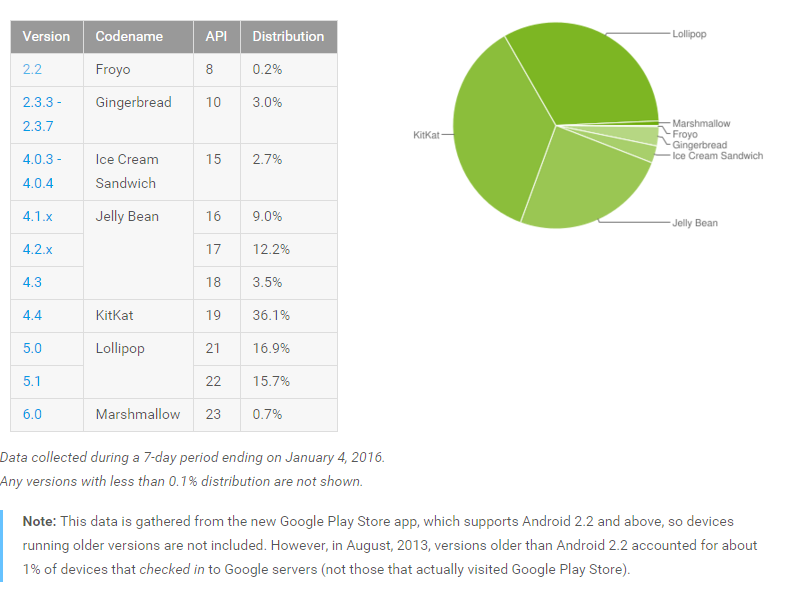
<http://developer.android.com/index.html>

<http://developer.android.com/about/dashboards/index.html>

**Last year**

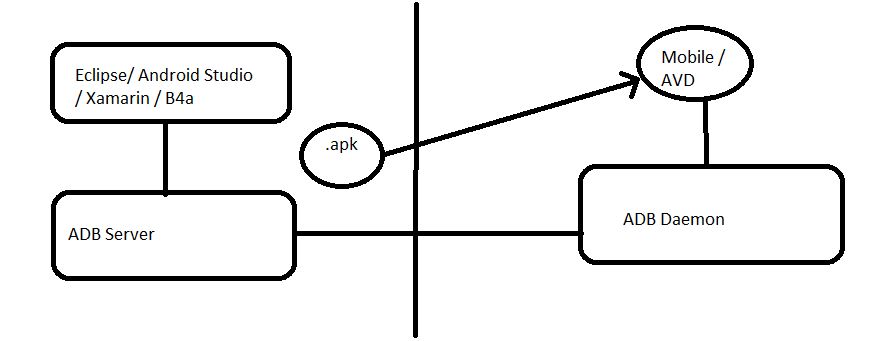


Current



**Programming in Android**

Android is built in such a way that you can work with it in different environments (flexible and not tied to a single development environment).



**Android Debug Bridge (ADB)**

It makes the communication between the desktop and mobile devices possible.

Finding the ADB command line utility in the Android folder

<install-folder>\sdk\platform-tools

* adb devices command

Shows the list of devices connected (actual devices and AVD’s)

* If things are not working correctly then you can always do the

adb kill-server command

* You can then run any adb command or use the start-server command to start the server

Xamarin Platform

Xamarin apps share code across all platforms.

Target iOS, Android, Windows and Mac with a single, shared C# codebase. Use the same language, APIs and data structures on every platform.

C# is the best language for mobile app development.

With Xamarin, you write your apps entirely in C#, sharing the same code on iOS, Android, Windows, Mac and more. Anything you can do in Objective-C, Swift or Java, you can do in C#.

Native UI, native API access & native performance.

Xamarin apps are built with standard, native user interface controls. Apps not only look the way the end user expects, they behave that way too. This can’t be achieved with other solutions.

**Installation**

1. Java JDK 7
2. Android SDK
3. Eclipse / Android Studio / Xamarin Studio
4. Configuring the IDE

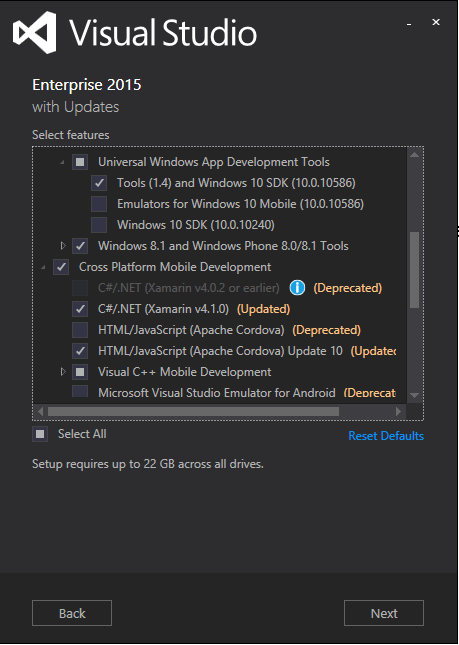
**Android Studio (uses Java)**

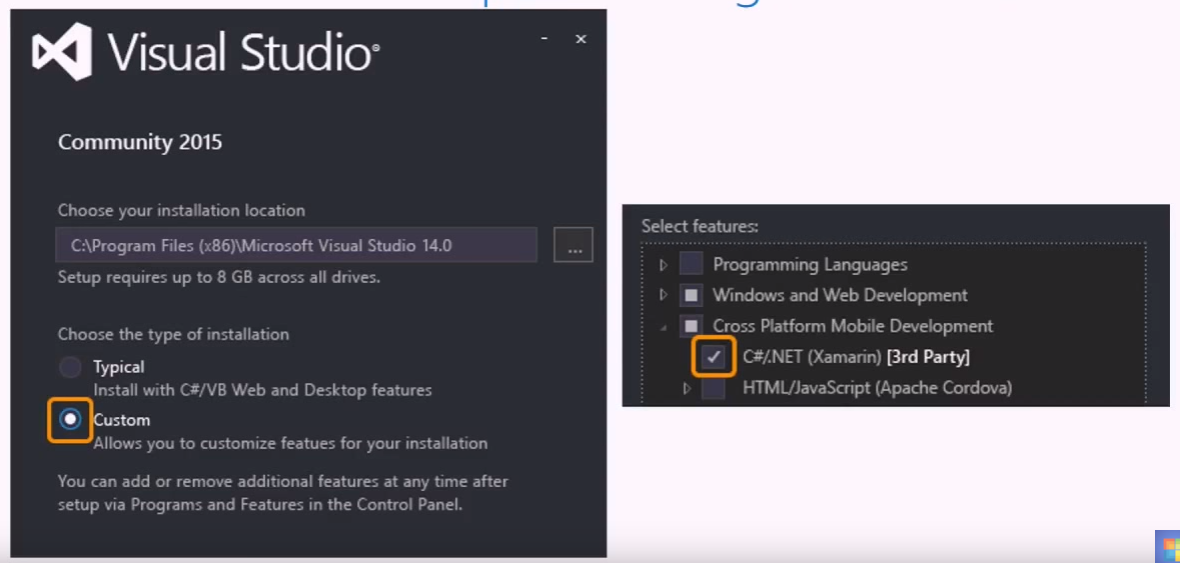
<http://developer.android.com/sdk/index.html>

**We will be using Visual Studio with Xamarin (C#.net)**

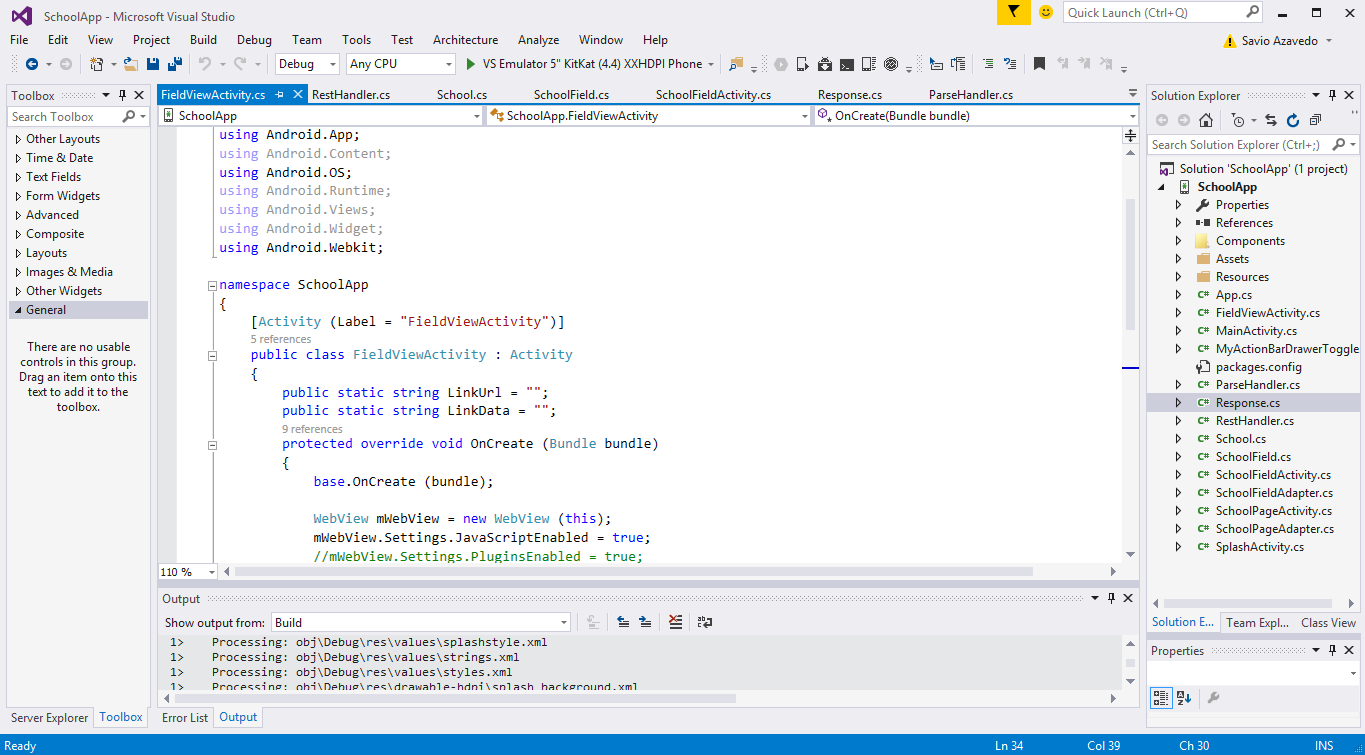
Since Microsoft purchased Xamarin, Xamarin is now freely available as part of the Visual Studio Community License. Xamarin is also available in other versions of Visual Studio.

If you have already installed Visual Studio or are running a new install, run the VS Installer and select Xamarin (C#.net) under cross-platform app development. This will automatically install Android SDK and the required components.



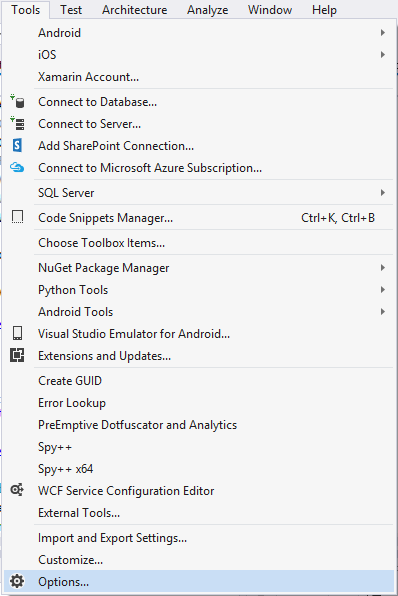


**Configuring Visual Studio**

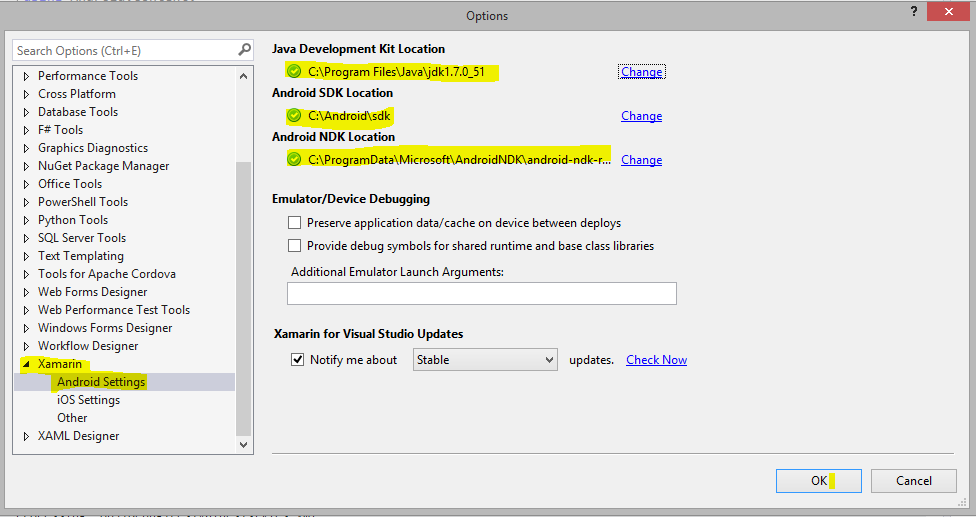


1. Checking your SDK paths

In VS go to Tools -> Options

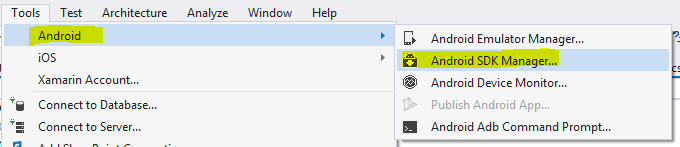


Select Xamarin -> Android Settings on the left and check your JDK, SDK and NDK paths. If the path is missing then you may have to manually point it to the correct folder.

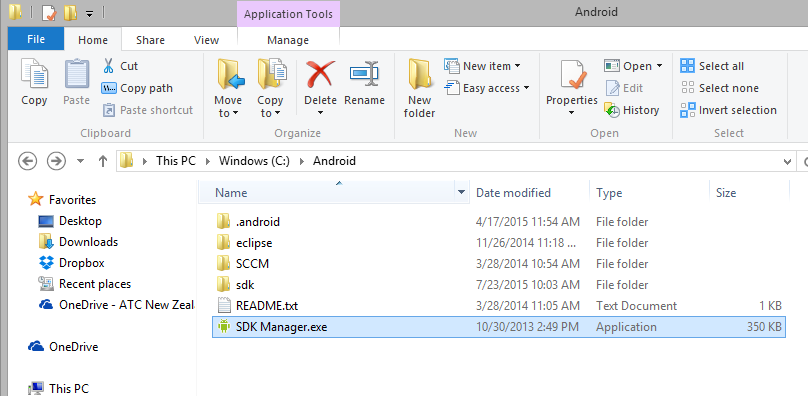


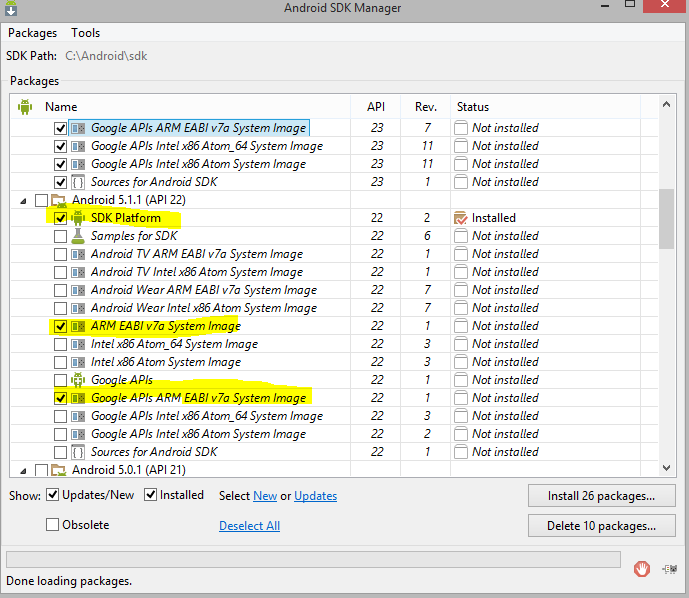
1. Android SDK Manager

Go to tools - > Android -> Android SDK manager



If it’s prompting for an admin password or does not work then you may need to go to the Android SDK folder and access it.





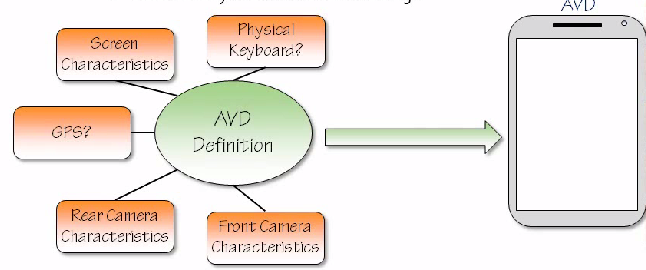
Based on the Android Platform you are developing for, you will have to install the respective SDK. As a minimum you will need the SDK platform, the ARM and Google API’s ARM system Images for the emulators.

1. Debugging and testing your app

**Android Virtual Devices (AVD) or emulator**

* Allows you to test your app on a desktop
* Android devices highly vary
* Lot of different target environments
* Create an AVD definition to describe the characteristics of a device
* Screen characteristics
* Physical keyboard
* GPS
* Camera

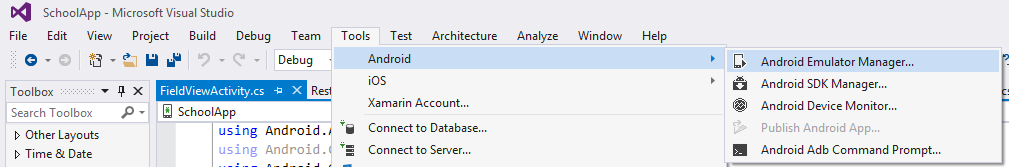
Create an AVD image to work with based on a AVD definition



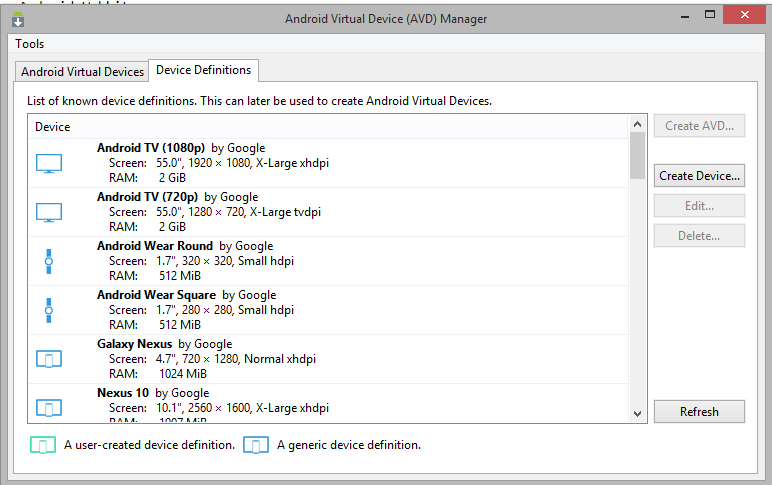
Changing the definition after the image has been created does not change the image.

We still need a device but not every possible device is needed. Emulators come in handy for testing on different environments and screen sizes.

Creating and accessing an AVD.

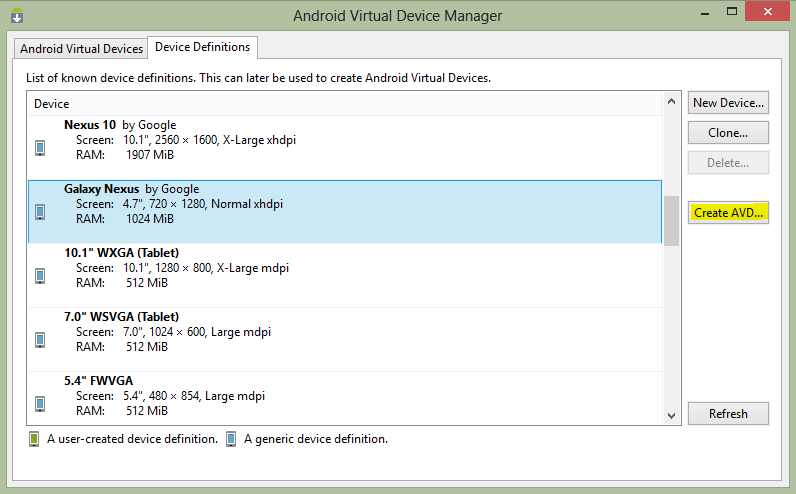


Go to Tools -> Android -> Android Emulator Manager



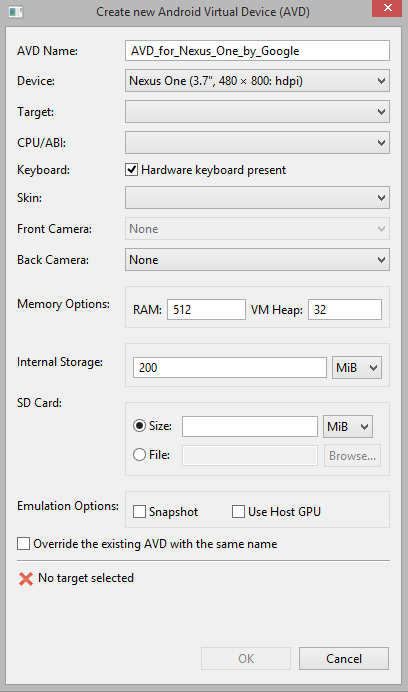
AVD’s can have device definitions based on

* Real device
* Generic device definitions



* Choose your destination device
* Click Create AVD
* Set the options and feature and click OK

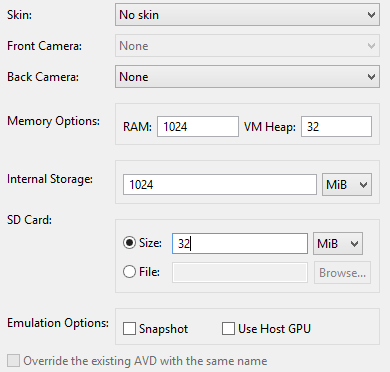
Starting the Android Virtual Device



Target - > Android version you want on the emulator

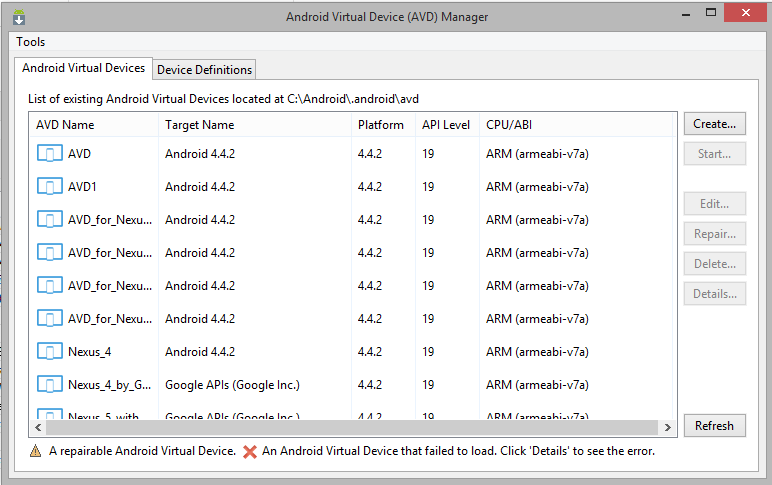
CPU/ABI -> Select ARM

Refer to the screen below for the rest of the options

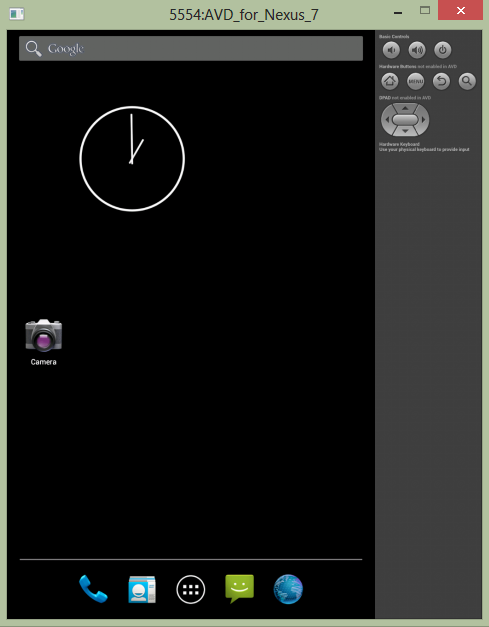


Press OK to create the AVD

After creating the device, under the Android Virtual Devices Tab you should be able to see your device.



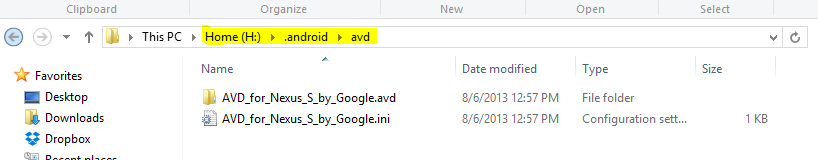
Select your device and press Start. It may take a long time to start for the first time. The emulator then starts up as shown.



**Troubleshooting**

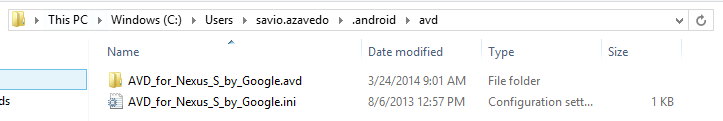
If you encounter the PANIC error while starting the AVD, then follow these steps to resolve it.

Go to the following location



Copy the newly created AVD folder (.avd) and the ini file

Paste it under your user’s directory in C:\ drive



Restart your Emulator

**Connecting an Actual Device**

AVD’s are not exactly the perfect replication of a real device.

* Some features are difficult to simulate such as the performance speed, GPS behaves differently.

Debugging on a real device requires extra setup

* Enable USB debugging on phone
* Install USB drivers on your desktop computer

Hooking the mobile and computer together is called Tethering



Phones by default do not have USB debugging enabled

Locating the USB debugging checkbox on your Android phone

< 4.0 Settings -> Applications -> Debugging

4.0, 4.1 Settings -> Developer Options

> 4.2

Developer Option is hidden

Settings -> About -> Build Number **(Tap the build number 7** **times)**

It becomes available in the previous screen

The USB driver enables development tools to communicate with device

* Most Android Developer Phones use the Google USB driver

(Nexus 1 and Nexus S) (Available through SDK manager)

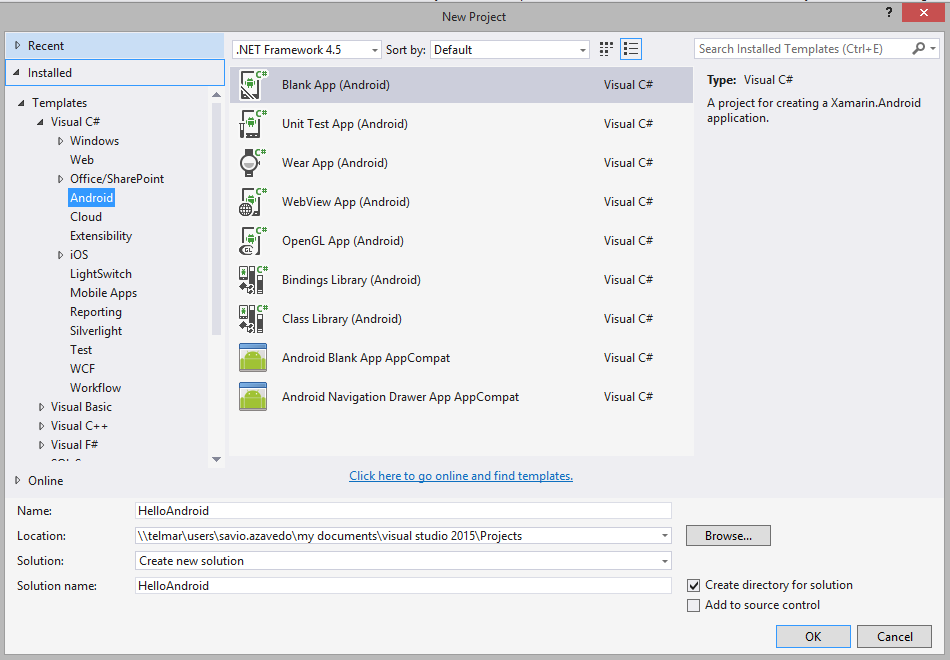
* Galaxy Nexus relies on Samsung provided driver
* For all other phones you must download the manufacturer drivers

<http://developer.android.com/tools/extras/oem-usb.html>

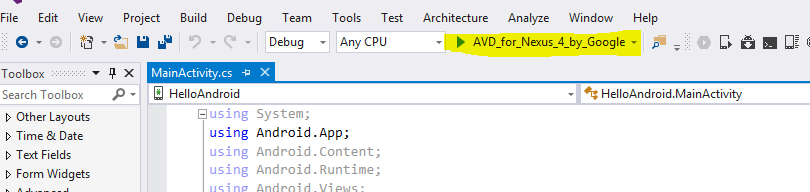
**First Hello World App**

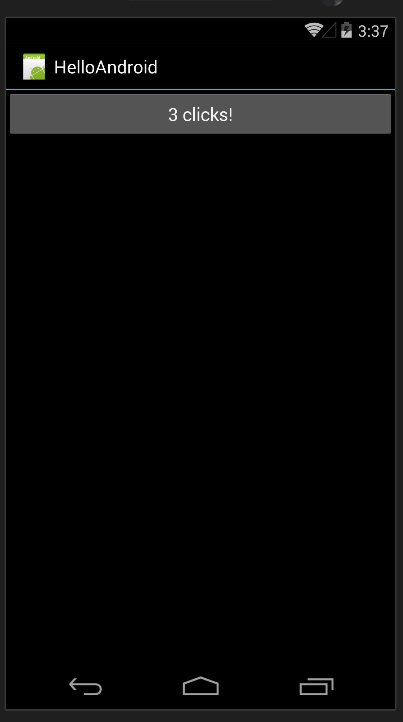
In VS select File -> New Project

On the left hand side of the dialog select Android under Visual C# and select Blank Android App on the right.



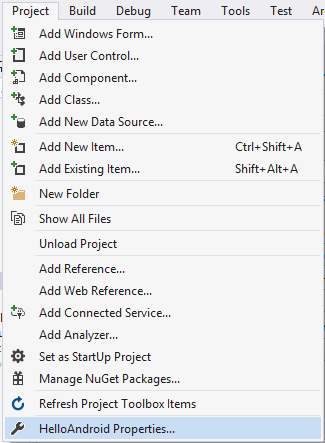
Select your emulator and run the program

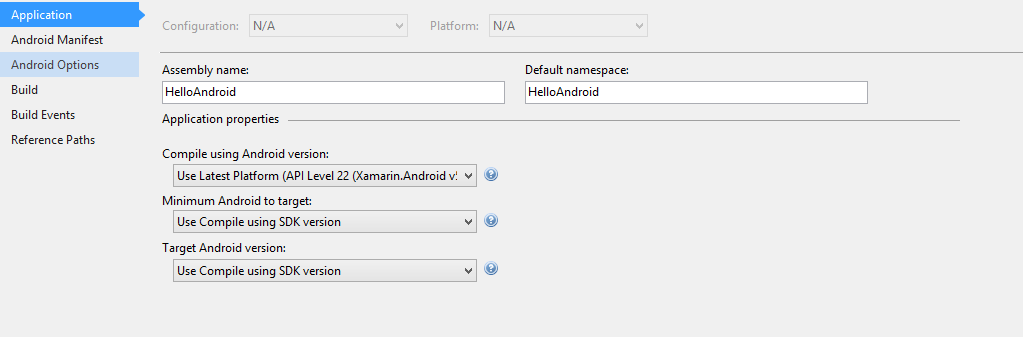




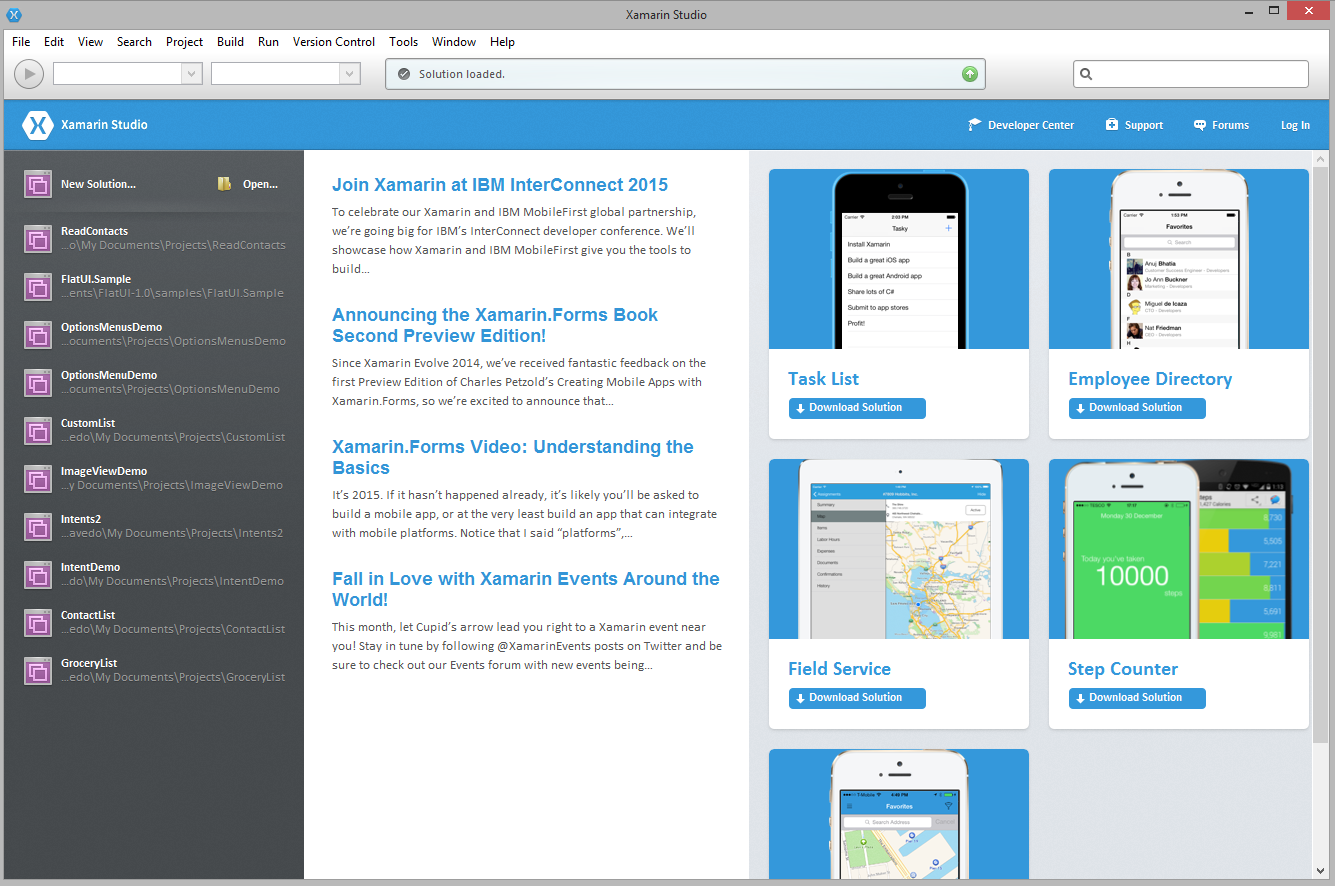
**Specifying an SDK level for an app**

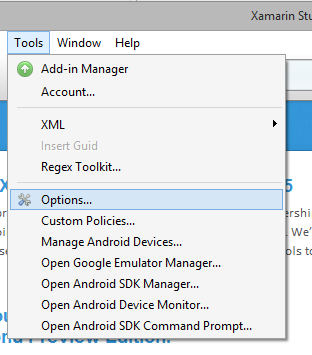
Go to Project -> Properties



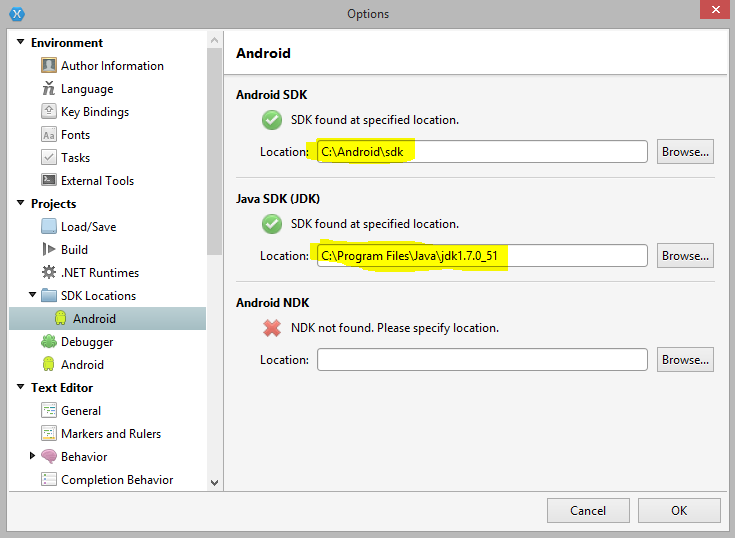


**Configuring Xamarin Studio**



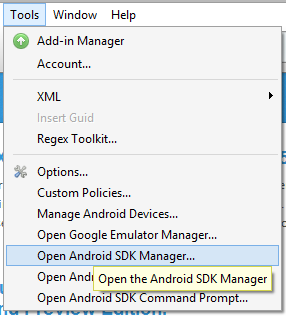


See that you’re SDK paths are properly pointing to the right location. It should install properly using the Universal Installer, however if you’re installing it separate then you may have to configure it manually.

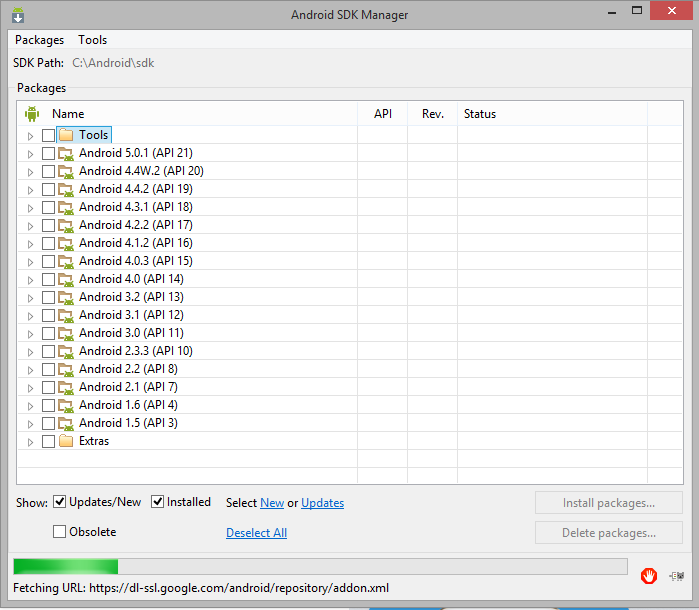


**Android SDK Manager**

Helps you install newer version of the Android SDK and the tools associated with it.

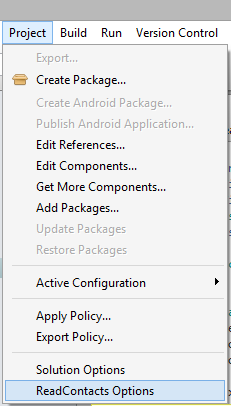


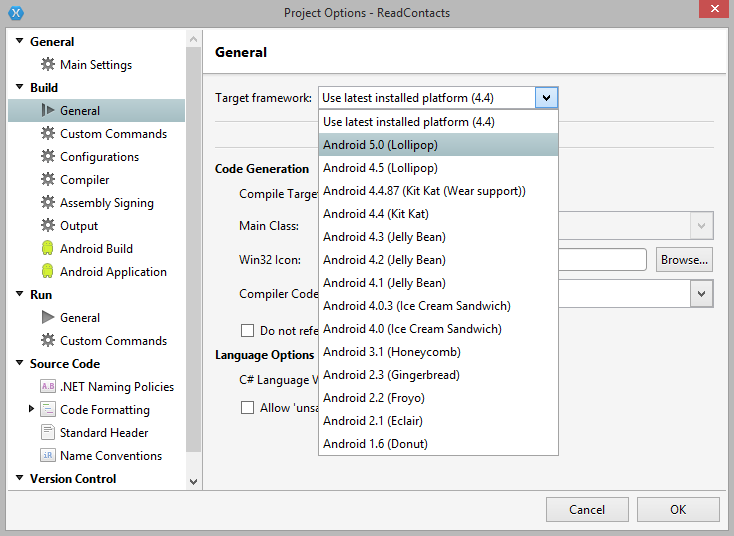
Android SDK manager.



**Specifying an SDK level for an App in Xamarin Studio**

Go to Project - > (Project Name) Options



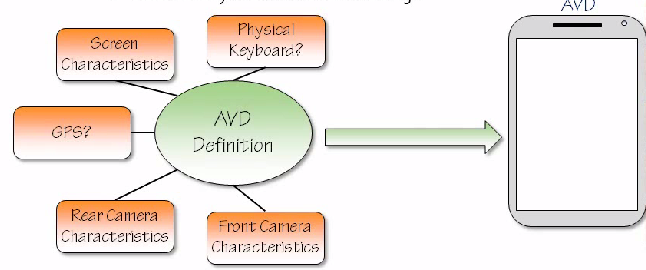
Select the android version to support.

**Debugging and Testing your App**

Android Virtual Devices (AVD) or emulator

* Allows you to test your app on a desktop
* Android devices highly vary
* Lot of different target environments
* Create an AVD definition to describe the characteristics of a device
* Screen characteristics
* Physical keyboard
* GPS
* Camera

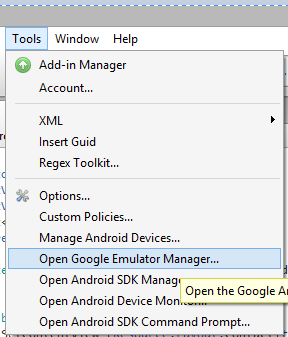
Create an AVD image to work with based on a AVD definition

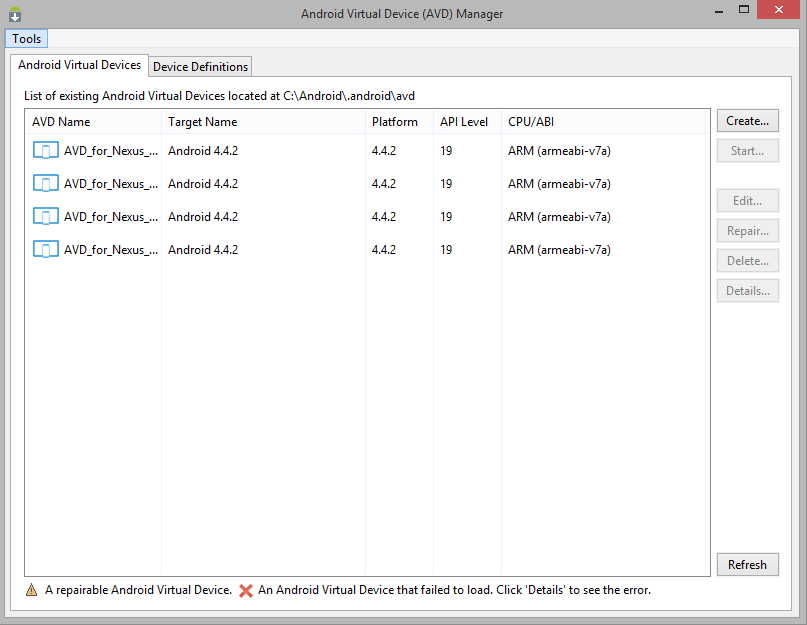


Changing the definition after the image has been created does not change the image.

We still need a device but not every possible device is needed. Emulators come in handy for testing on different environments and screen sizes.

Accessing the AVD.





AVD’s can have device definitions based on

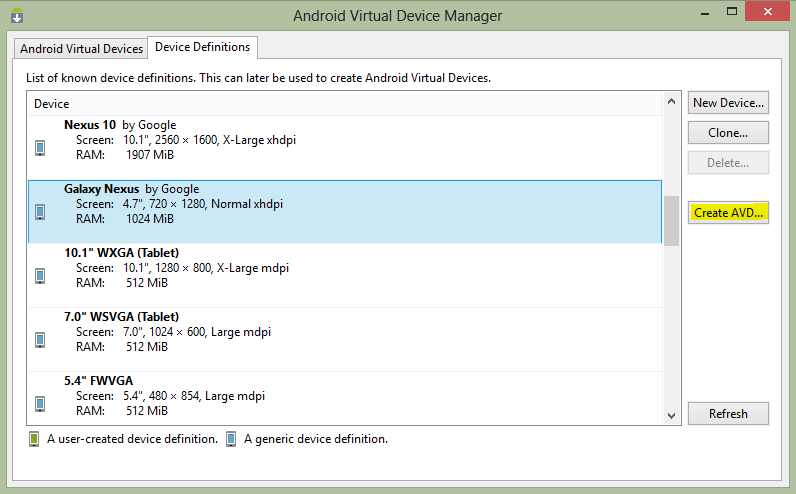
* Real device
* Generic device definitions

Real Device



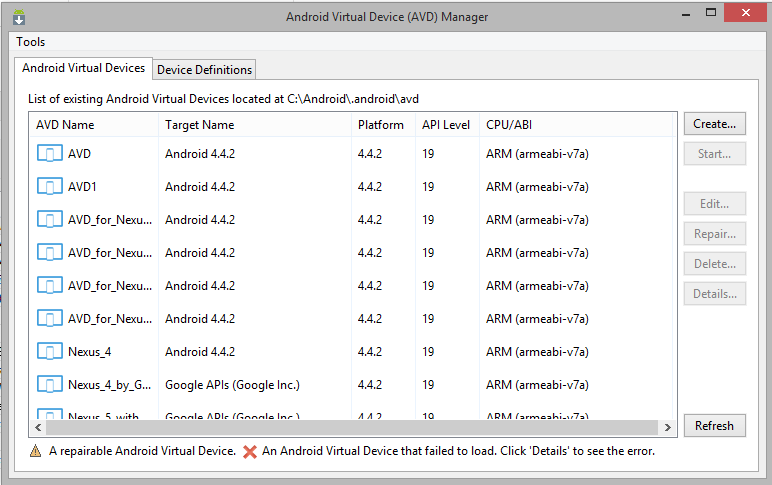
Generic Device Definitions





* Choose your destination device
* Click Create AVD
* Set the options and feature and click OK

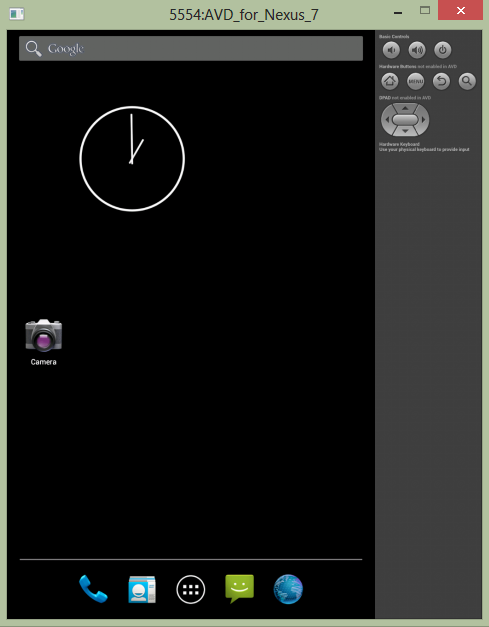
Starting the Android Virtual Device



Click on a device and press start.

It may take a long time to start for the first time.

The emulator then starts up as shown.



**Set the “Stay Awake” option**

The device going to sleep can be disturbing while debugging.

Set it in the developer options or download an app from the playstore.

**DDMS (Dalvik Debug Monitoring System)**

Android tool that provides a whole bunch of debugging features.

sdk\ tools

