Lab – Android Development Environment

Setting up the ADT, Creating, Running and Debugging Your First Application

Objectives:

Familiarize yourself with the Android Development Environment

Important Note: This class has many students with a wide range of previous experience. Some students are fairly new to object-oriented programming (OOP). Some have OOP experience, but are new to Android. Still others have some Android experience already, and want to just freshen up their knowledge.

Because of this, I'm not expecting that everyone can finish this entire lab. I suggest that you set a time limit for yourself, say 1 hour. Work through what you can in that time and then stop and take a break. If you later feel that you have some more time for this Lab, then repeat the process. Again – don't feel that you need to finish everything in this lab. That's not the goal here.

Specifically, if you are fairly new to programming, you should try to complete Parts 1 – 4 below. If you are familiar with programming and programming environments, you should try to complete parts 1 – 6 below.

This lab contains the following Parts.

1. Set up Android Studio. We have broken this up into 3 different sections, depending on what operating system you are running locally. 1a is for Mac users, 1b is for Windows users and 1c is for Linux users. You only need to read through and follow the instructions for the operating system you are running on your development machine.
2. Create a new Android application.
3. Create an Android Virtual Device and start the Android Emulator.
4. Run the application you created in Part 2.
5. Import an application project.
6. Debug an Android application.

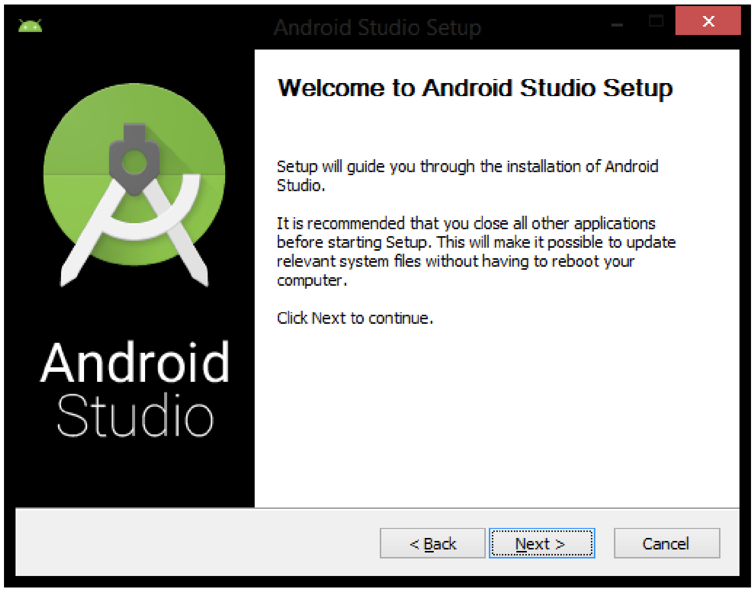
Additional helpful information can be found on the Android Developer website:

- https://developer.android.com/studio/index.html
- https://developer.android.com/training/basics/firstapp/creating-project.html
- https://developer.android.com/studio/run/managing-avds.html
- https://developer.android.com/training/basics/firstapp/running-app.html

Part 1a – Setting Up Android Studio on OSX (Mac) .

In this part you will download and install Android Studio which will be the Integrated Development Environment (IDE) used for this course. For the purposes of this document, we installed Android Studio version 4.0.1 (the current latest stable release as of 6/24/2020) on a Mac running Catalina (10.15.6). All screenshots correspond to that environment.

1. Download the current version of android studio 4.0.1 for Mac from https://developer.android.com/studio
2. Click on 'Download Android Studio'.
3. Open the executable file android-studio-<xxx>.
4. Once the setup loads, you will see the Welcome Screen.



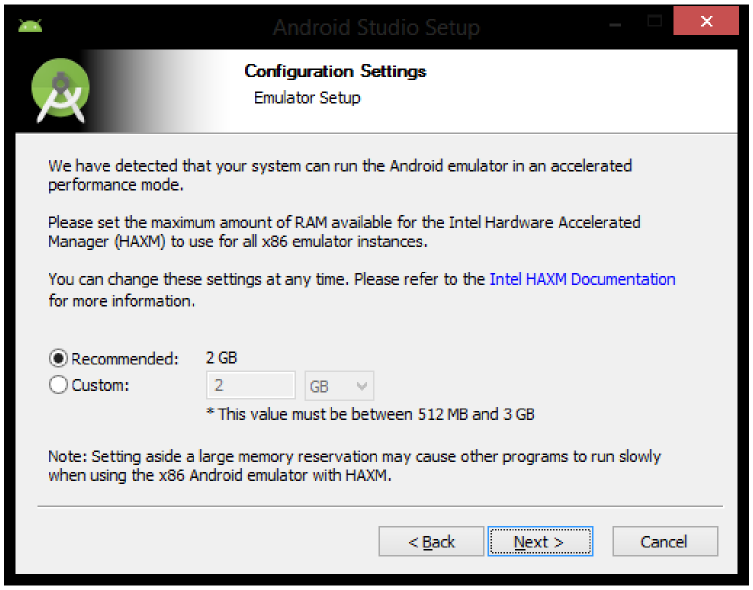
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

5. Click 'Next >' on the Welcome Screen.
6. When choosing components, ensure all of the checkboxes are checked in for each component to install. Once you are done, click 'Next >'.
7. Agree to the Android Studio and the Intel HAXM License Agreements after reading them.
8. Verify the install locations meet the installation requirements and click 'Next >'.
9. You may or may not see the emulator setup settings, just click 'Next >' after selecting the RAM size.



We have detected that your system can run the Android emulator in an accelerated performance mode.

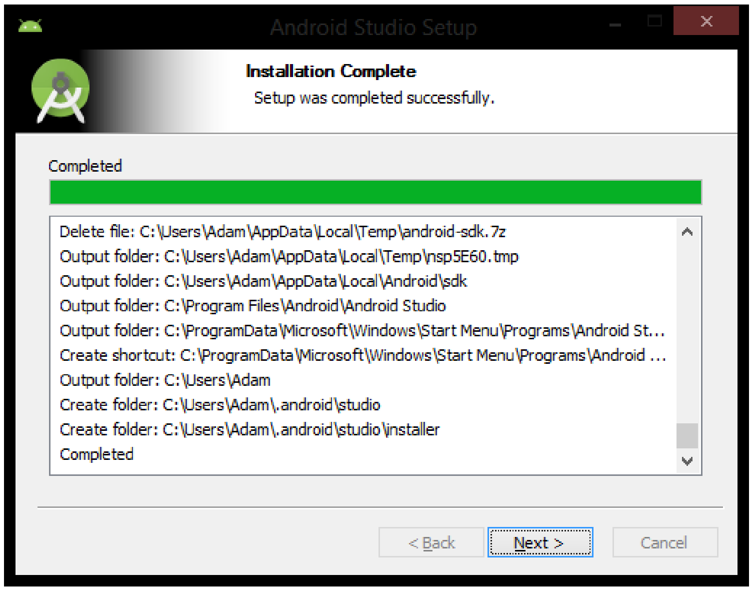
Please set the maximum amount of RAM available for the Intel Hardware Accelerated Manager (HAXM) to use for all x86 emulator instances.

You can change these settings at any time. Please refer to the Intel HAXM Documentation for more information.

● Recommended: 2 GB
○ Custom: 2 GB
* This value must be between 512 MB and 3 GB

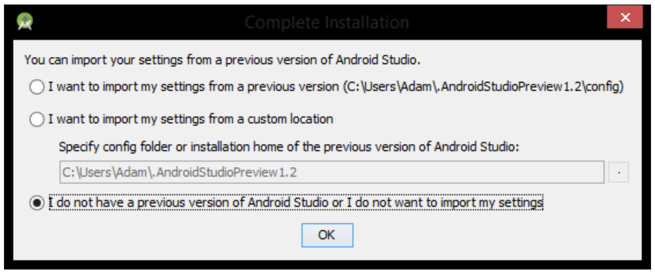
Note: Setting aside a large memory reservation may cause other programs to run slowly when using the x86 Android emulator with HAXM.

10. Finally, click ‘Install’. You will see which operations are currently running in the installation process and a progress bar displaying their progress.
11. Once the installation process is finished click ‘Next >’.



```
Delete file: C:\Users\Adam\AppData\Local\Temp\android-sdk.7z
Output folder: C:\Users\Adam\AppData\Local\Temp\nsp5E60.tmp
Output folder: C:\Users\Adam\AppData\Local\Android\sdk
Output folder: C:\Program Files\Android\Android Studio
Output folder: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Android St...
Create shortcut: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Android ...
Output folder: C:\Users\Adam
Create folder: C:\Users\Adam\.android\studio
Create folder: C:\Users\Adam\.android\studio\installer
Completed
```

12. Android Studio is now set up. Check on ‘Start Android Studio’ and click ‘Finish’.
13. You will see the Complete Installation screen below.
14. If you had a previous version of Android Studio installed prior, you can choose either of the first 2 radio boxes to import settings from previous install. If you do not have a previous install or do not want to import settings from a previous install, select the third radio button.



You can import your settings from a previous version of Android Studio.

○ I want to import my settings from a previous version (C:\Users\Adam\.AndroidStudioPreview1.2\config)

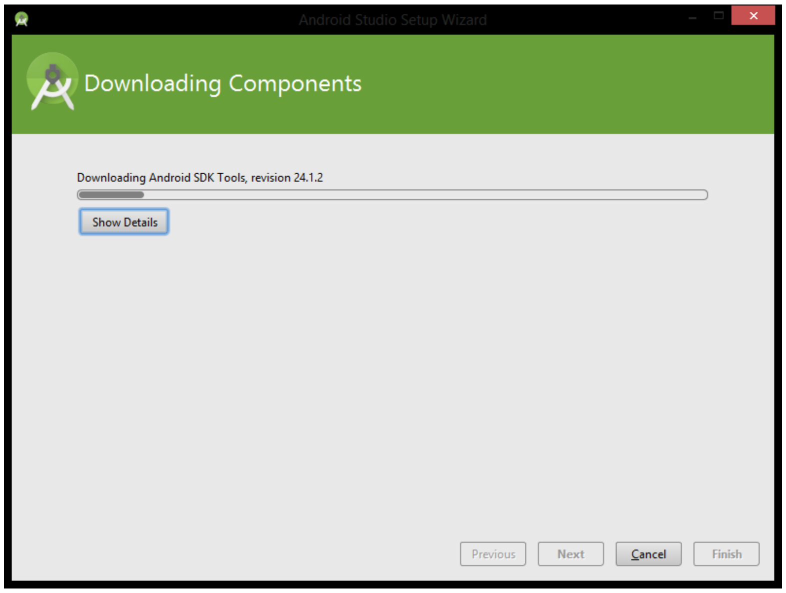
○ I want to import my settings from a custom location

Specify config folder or installation home of the previous version of Android Studio:

C:\Users\Adam\.AndroidStudioPreview1.2

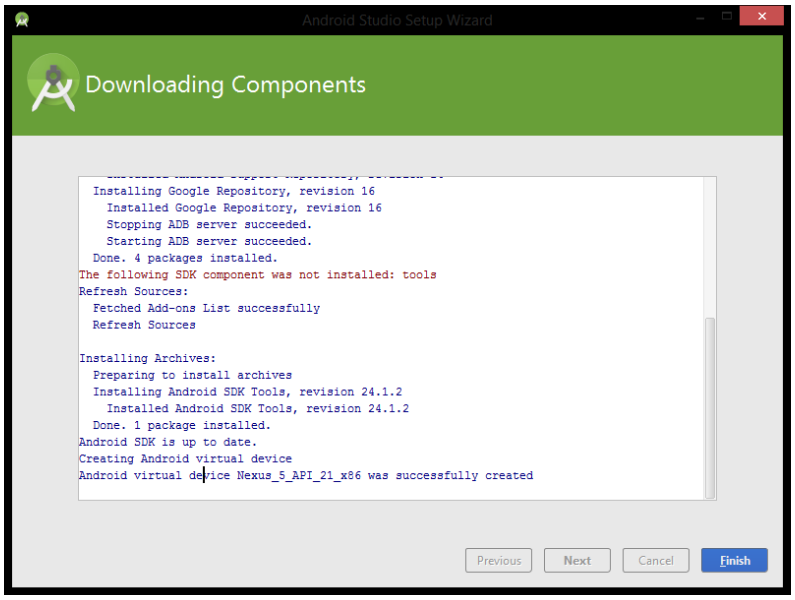
● I do not have a previous version of Android Studio or I do not want to import my settings

15. After the splash screen you may see some additional setup operations run, such as downloading components.



Downloading Android SDK Tools, revision 24.1.2

16. Once it is finished, click ‘Finish’.



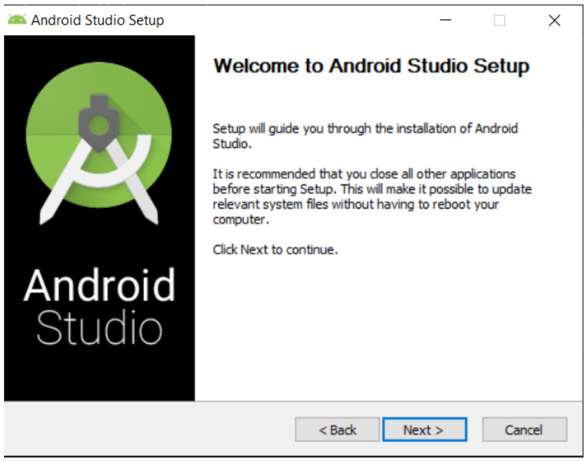
```
  Installing Google Repository, revision 16
    Installed Google Repository, revision 16
    Stopping ADB server succeeded.
    Starting ADB server succeeded.
  Done. 4 packages installed.
The following SDK component was not installed: tools
Refresh Sources:
  Fetched Add-ons List successfully
  Refresh Sources

Installing Archives:
  Preparing to install archives
  Installing Android SDK Tools, revision 24.1.2
    Installed Android SDK Tools, revision 24.1.2
  Done. 1 package installed.
Android SDK is up to date.
Creating Android virtual device
Android virtual device Nexus_5_API_21_x86 was successfully created
```

17. Welcome to Android Studio! In the next part we will start our first project.

Part 1b – Setting Up Android Studio on Windows.

In this part you will download and install Android Studio which will be the Integrated Development Environment (IDE) used for this course. For the purposes of this document, we installed Android Studio version 4.0.1 (the current latest stable release as of July 2020) on a Windows 64-bit machine running the Windows 10 OS. All screenshots correspond to that environment.

1. Download Android Studio from https://developer.android.com/studio/index.html . Click on the 'Download Android Studio' button.
2. Open the executable file android-studio-<xxx>. If asked for permissions to open the app, click on 'Yes'.
3. Once the setup loads, you will see the Welcome Screen.



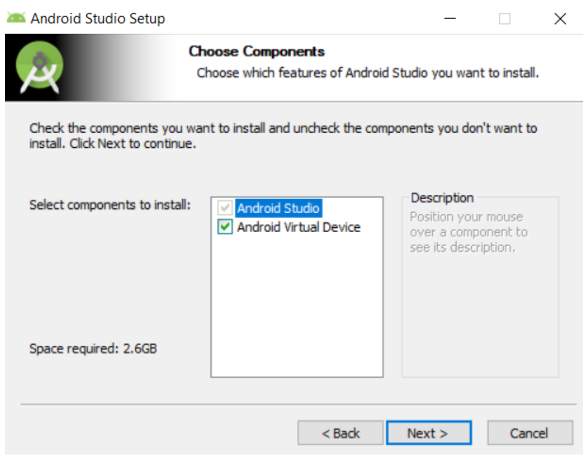
Welcome to Android Studio Setup

Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

4. Click 'Next >' on the Welcome Screen.
5. When choosing components, ensure all of the checkboxes are checked in for each component to install. Once you are done, click 'Next >'.



Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- [✓] Android Studio
- [✓] Android Virtual Device

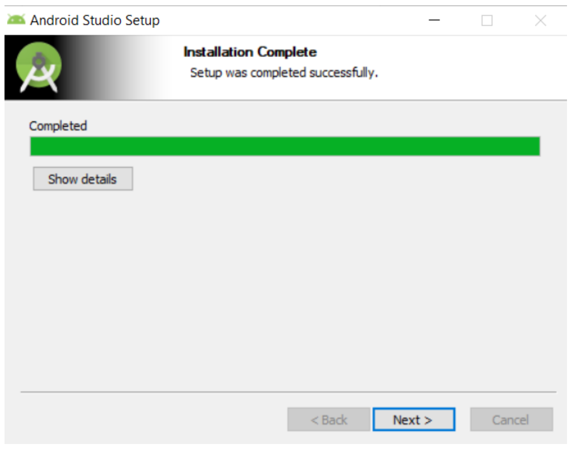
Description

Position your mouse over a component to see its description.

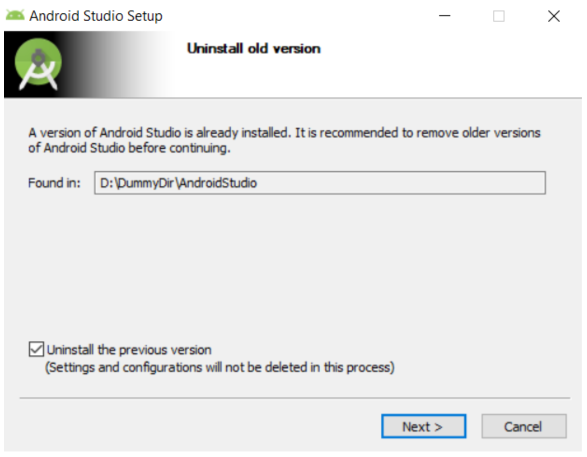
Space required: 2.6GB

6. Verify the install locations meet the installation requirements and click 'Next >'.

7. Select the start menu folder if required (Let it be Android Studio, it auto-populates). If you don't wish to create a shortcut, check the box 'Do not create shortcuts'.
8. Finally, click 'Install'. You will see which operations are currently running in the installation process and a progress bar displaying their progress.
9. Once the installation process is finished click 'Next >'.



10. Android Studio is now set up. Check on 'Start Android Studio' and click 'Finish'.
11. You will see the Complete Installation screen below.
12. If you had a previous version of Android Studio installed prior, Windows will recommend uninstalling the previous version and then reinstall the new version. If you do not wish to do so, you may uncheck the box and install in a different directory (not recommended).



A version of Android Studio is already installed. It is recommended to remove older versions of Android Studio before continuing.

Found in: D:\DummyDir\AndroidStudio

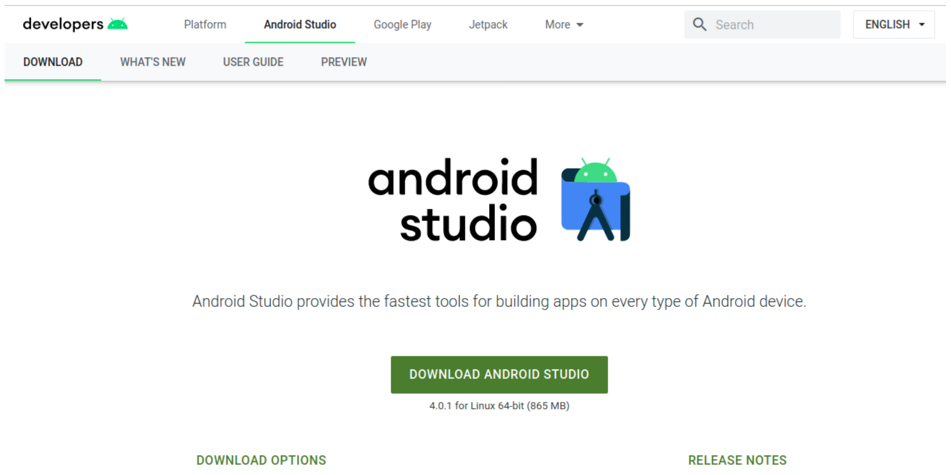
☑ Uninstall the previous version
(Settings and configurations will not be deleted in this process)

Welcome to Android Studio! In part 2 we will start our first project.

Part 1c – Setting Up Android Studio on a Linux Distribution (Ubuntu 18.04).

1. Download the current version of android studio 4.0.1 for Linux 64 bit from https://developer.android.com/studio

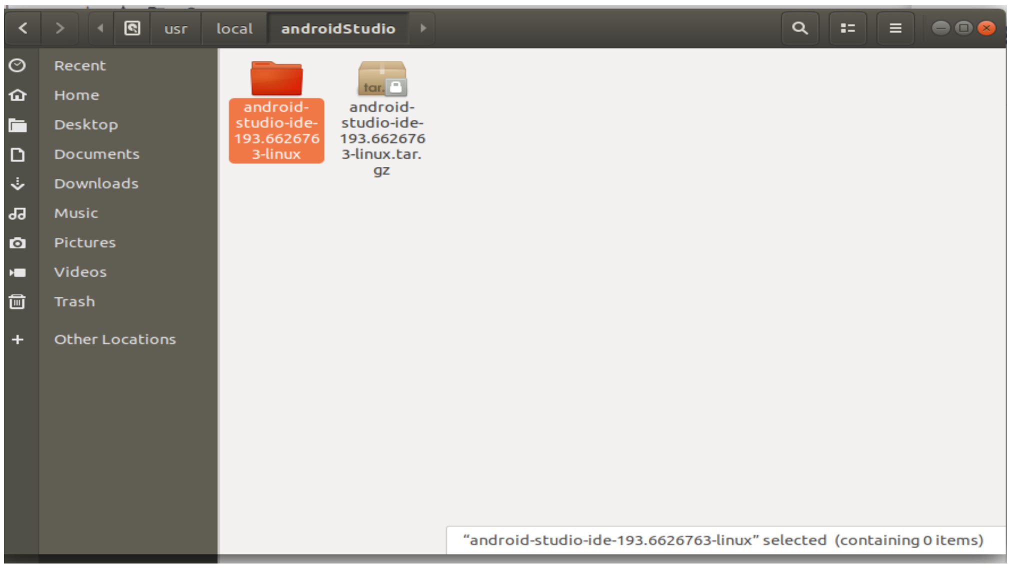
   Complete Installation guide: https://developer.android.com/studio/install#64bit-libs



Android Studio provides the fastest tools for building apps on every type of Android device.

4.0.1 for Linux 64-bit (865 MB)

2. Unzip the package to an appropriate location for your applications, such as within /usr/local/ for your user profile, or /opt/ for shared users. Note: you have to use sudo for making the directory and give permission to write, to the new folder where you extract android studio.

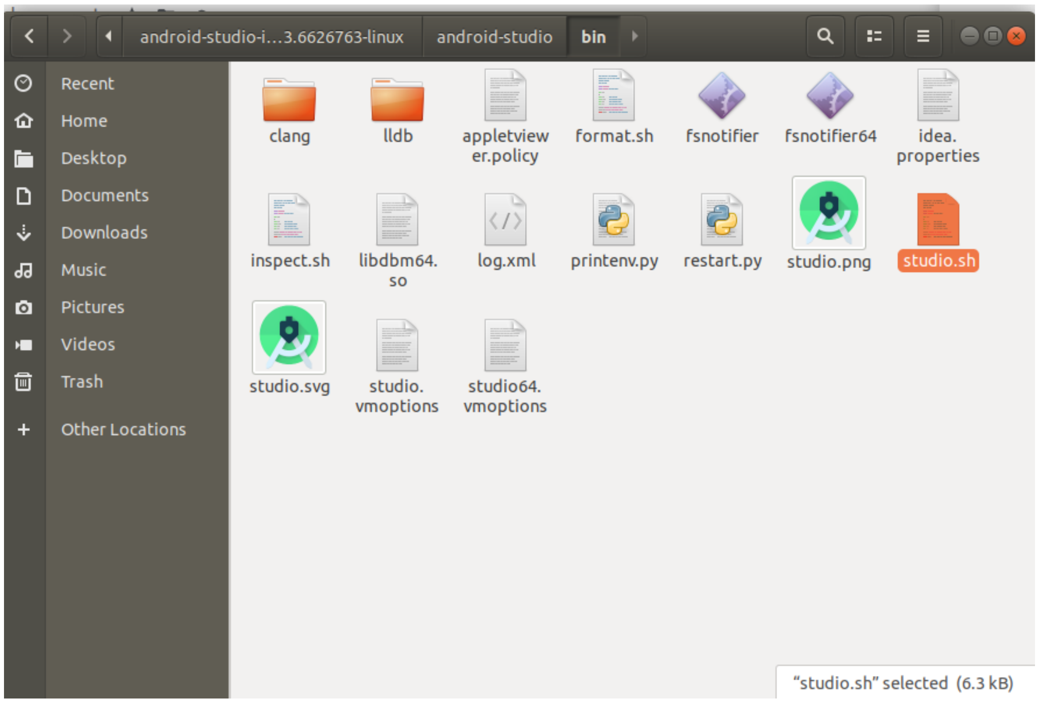


"android-studio-ide-193.6626763-linux" selected (containing 0 items)

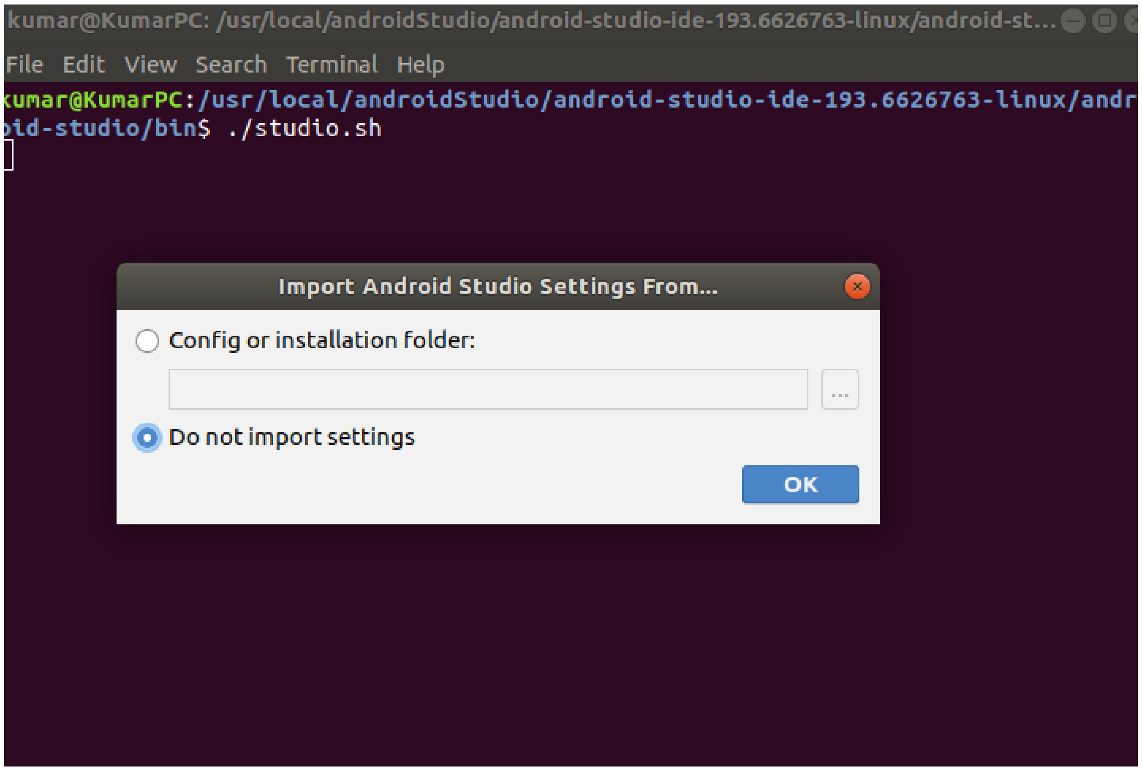
3. Please install required libraries by running this command :
   sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1 libbz2-1.0:i386

   These are 32 bit libraries required by the 64 bit version of Ubuntu.

4. To launch Android Studio, open a terminal, navigate to the **android-studio/bin/** directory, and execute studio.sh by typing **./studio.sh** command.



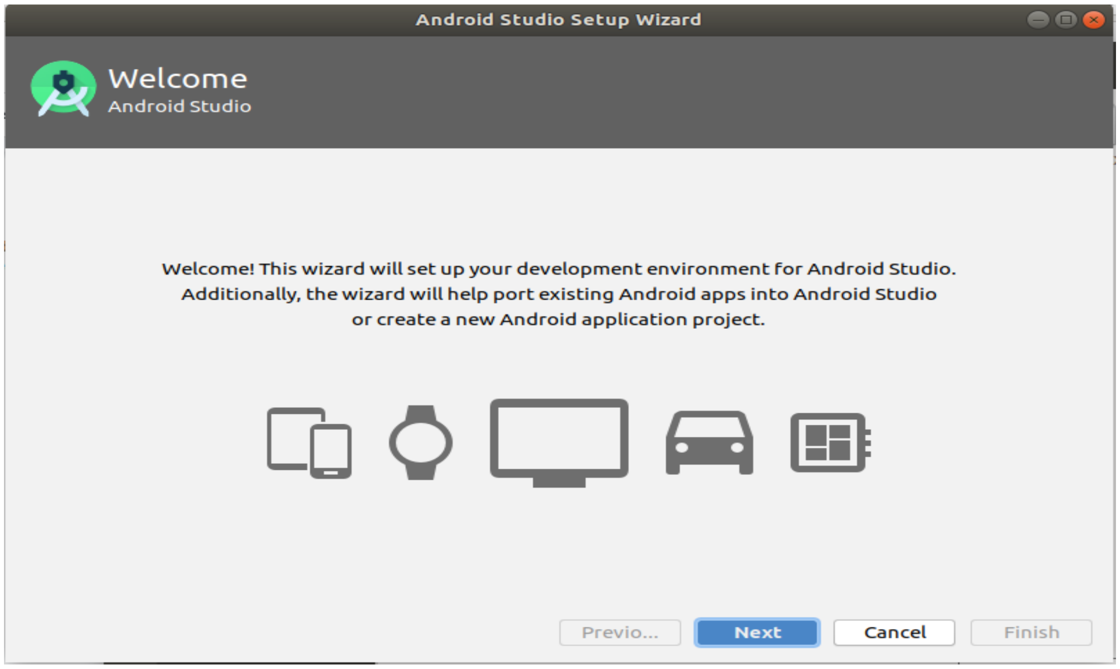
5. If you are installing it for the first time, select **do not import settings,** after running the above command/ script.


File Edit View Search Terminal Help
Import Android Studio Settings From…
OK

```
kumar@KumarPC:/usr/local/androidStudio/android-studio-ide-193.6626763-linux/andr
oid-studio/bin$ ./studio.sh
```

Config or installation folder: [✓ Do not import settings]

6. Click on Next.

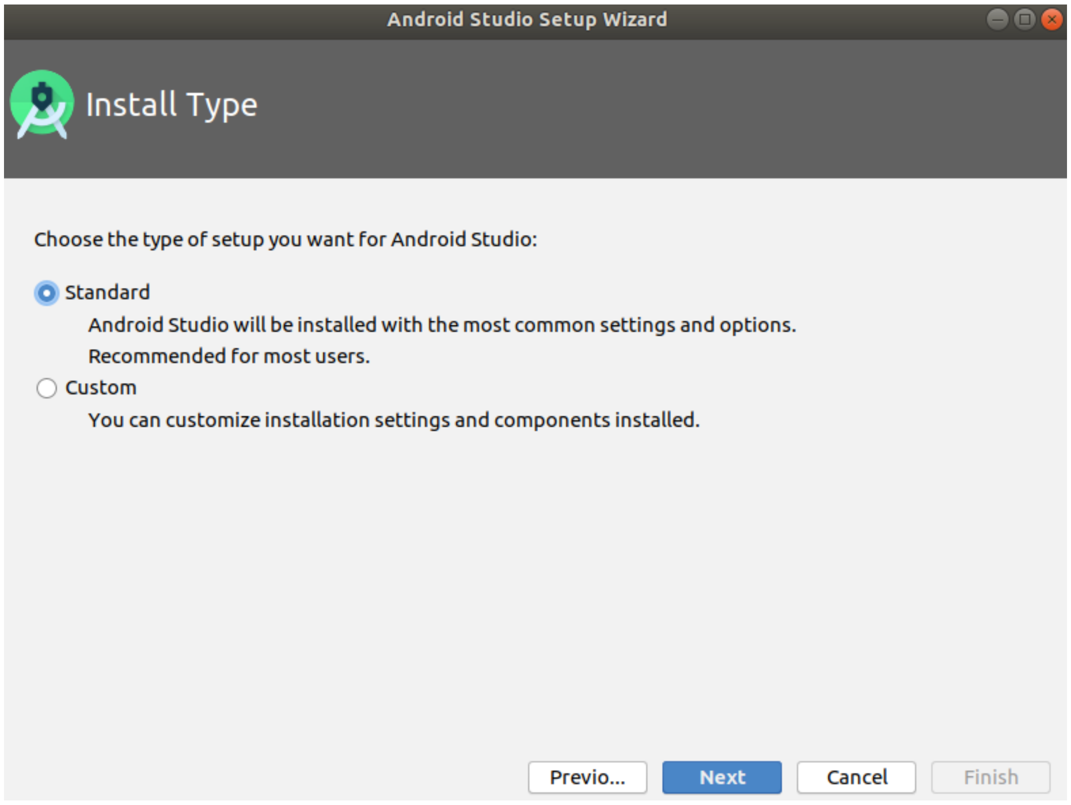


Android Studio Setup Wizard

Welcome
Android Studio

Welcome! This wizard will set up your development environment for Android Studio.
Additionally, the wizard will help port existing Android apps into Android Studio
or create a new Android application project.

7. Select Standard Installation.



Android Studio Setup Wizard

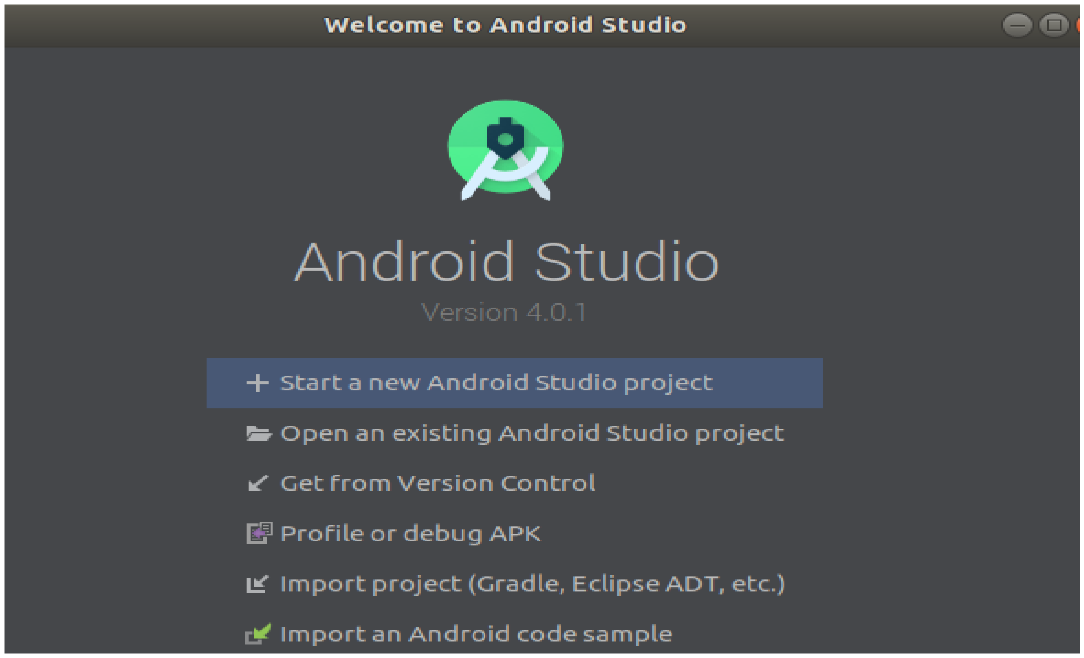
Install Type

Choose the type of setup you want for Android Studio:

[✓ Standard]
Android Studio will be installed with the most common settings and options.
Recommended for most users.

Custom
You can customize installation settings and components installed.

8. Android studio 4.0.1 is installed.


Welcome to Android Studio
Android Studio
Version 4.0.1
+ Start a new Android Studio project
Open an existing Android Studio project
Get from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample

Welcome to Android Studio! In part 2 we will start our first project.

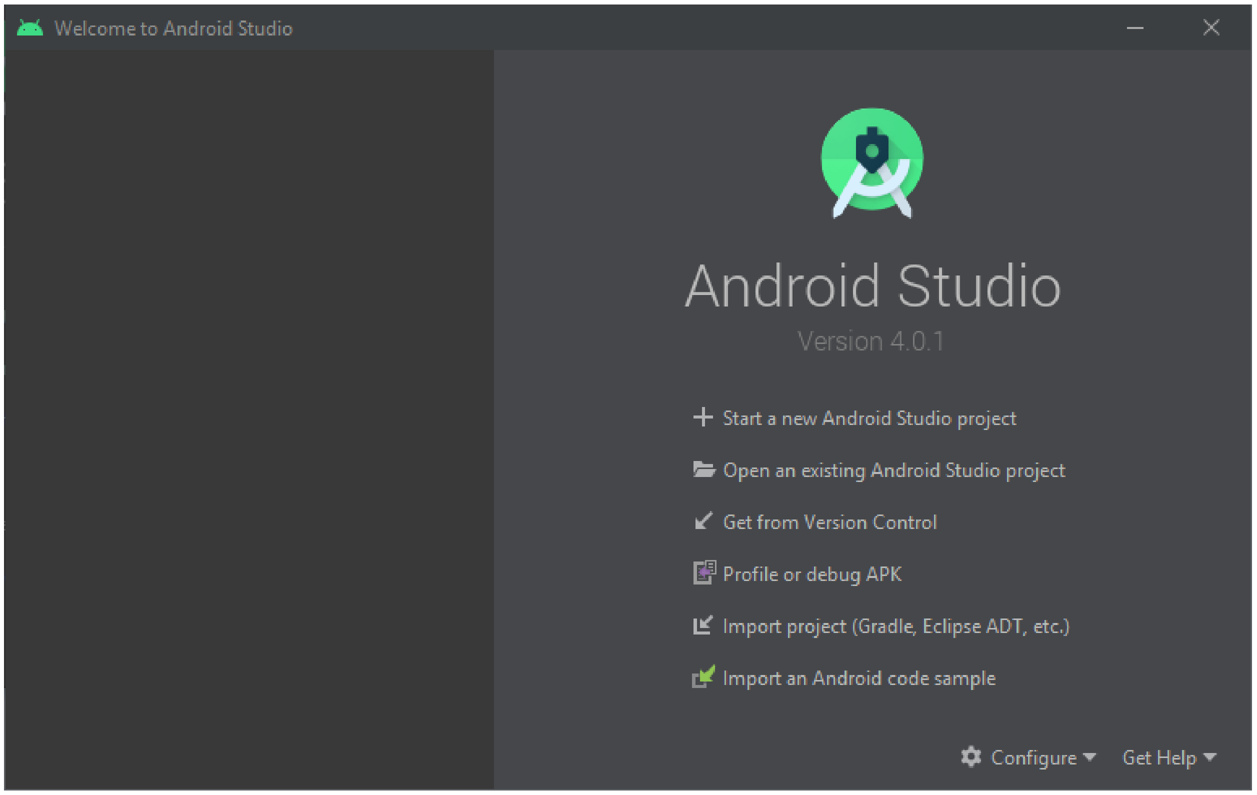
Part 2 – Creating a New Project

In this part, you will create a simple Android application that displays the words, “Hello World!”.

1. When you first launch Android Studio, you will be met with this screen.

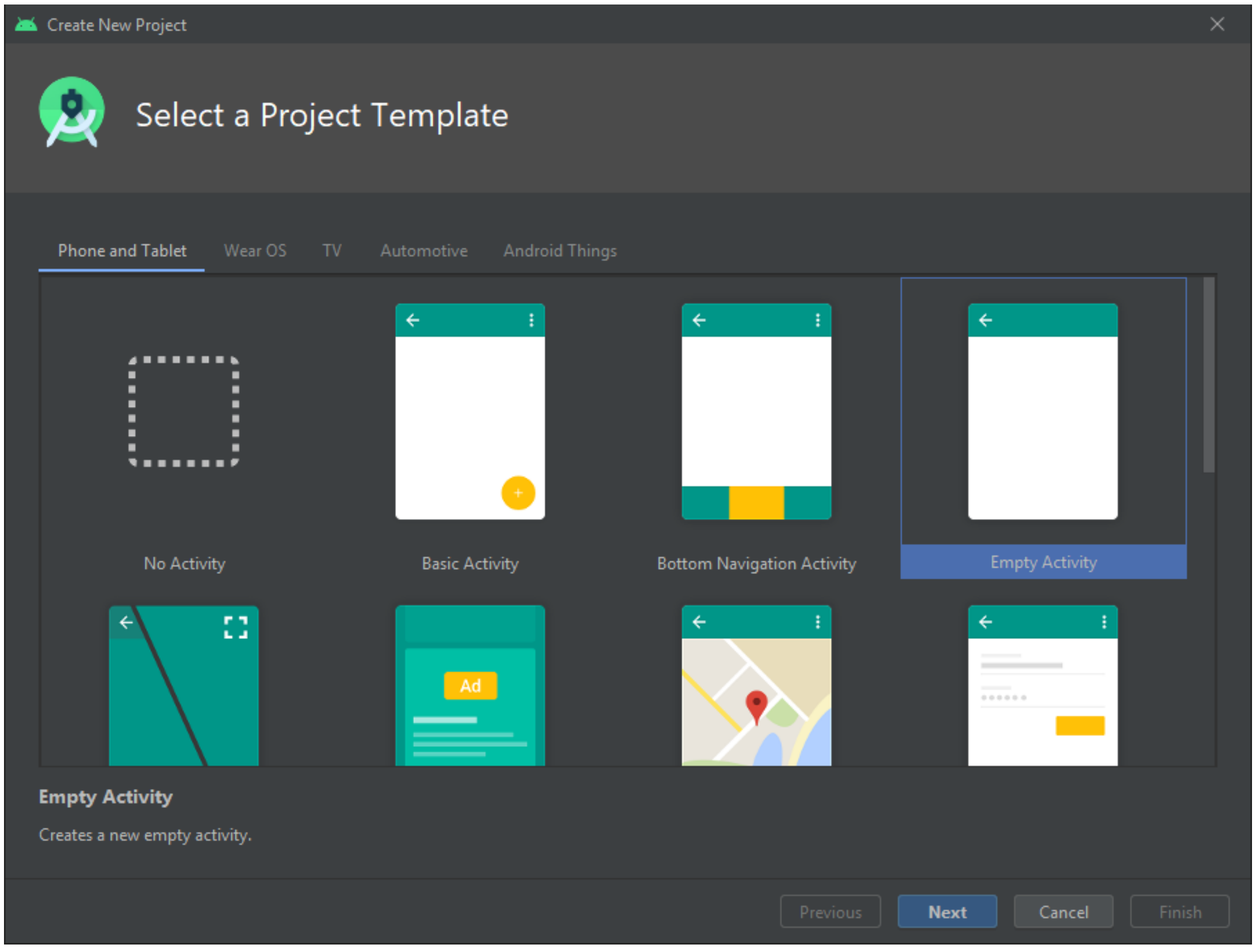
   Start a new Android Studio Project.

   Note: You may not have a side pane as shown in this image.


Welcome to Android Studio
Android Studio
Version 4.0.1
+ Start a new Android Studio project
Open an existing Android Studio project
Get from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

2. Create a new ‘Empty Activity’.

   For the purposes of this lab, we will be starting from an Empty Activity.



Empty Activity

Creates a new empty activity.

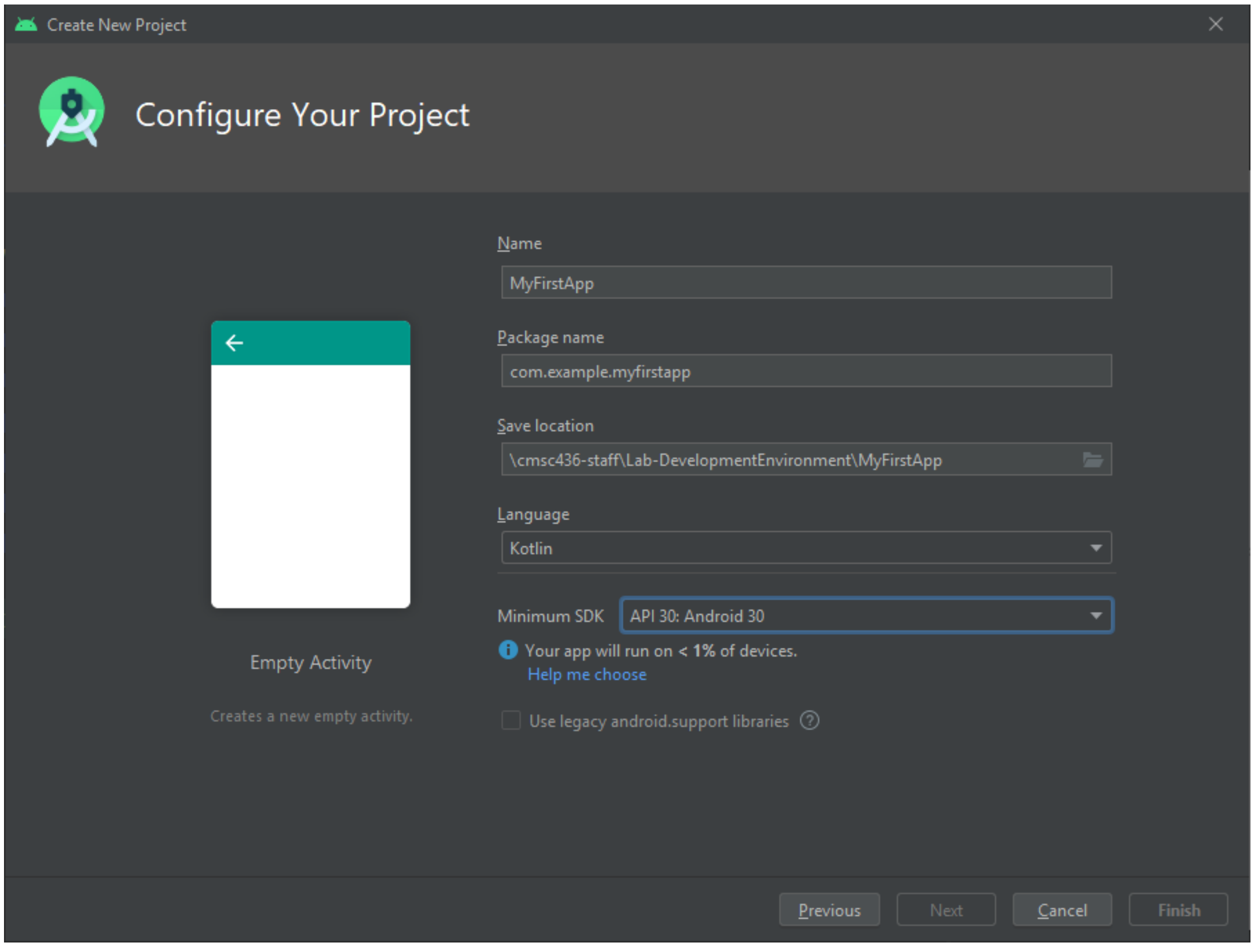
3. Name the app ‘MyFirstApp’.

   The package name is not significant for this class, but the general naming convention is to have a URL backwards.

   Select where you would like the project to be stored on your file system.
   Note: I am storing the project inside the Lab-DevelopmentEnvironment directory of the cloned 436 repo on my local system.

   Make sure you select ‘Kotlin’ as the language and set the minimum API Level to 30.

   Android should now create and build your new project.



Create New Project

Configure Your Project

Name: MyFirstApp

Package name: com.example.myfirstapp

Save location: \cmsc436-staff\Lab-DevelopmentEnvironment\MyFirstApp

Language: Kotlin

Minimum SDK: API 30: Android 30

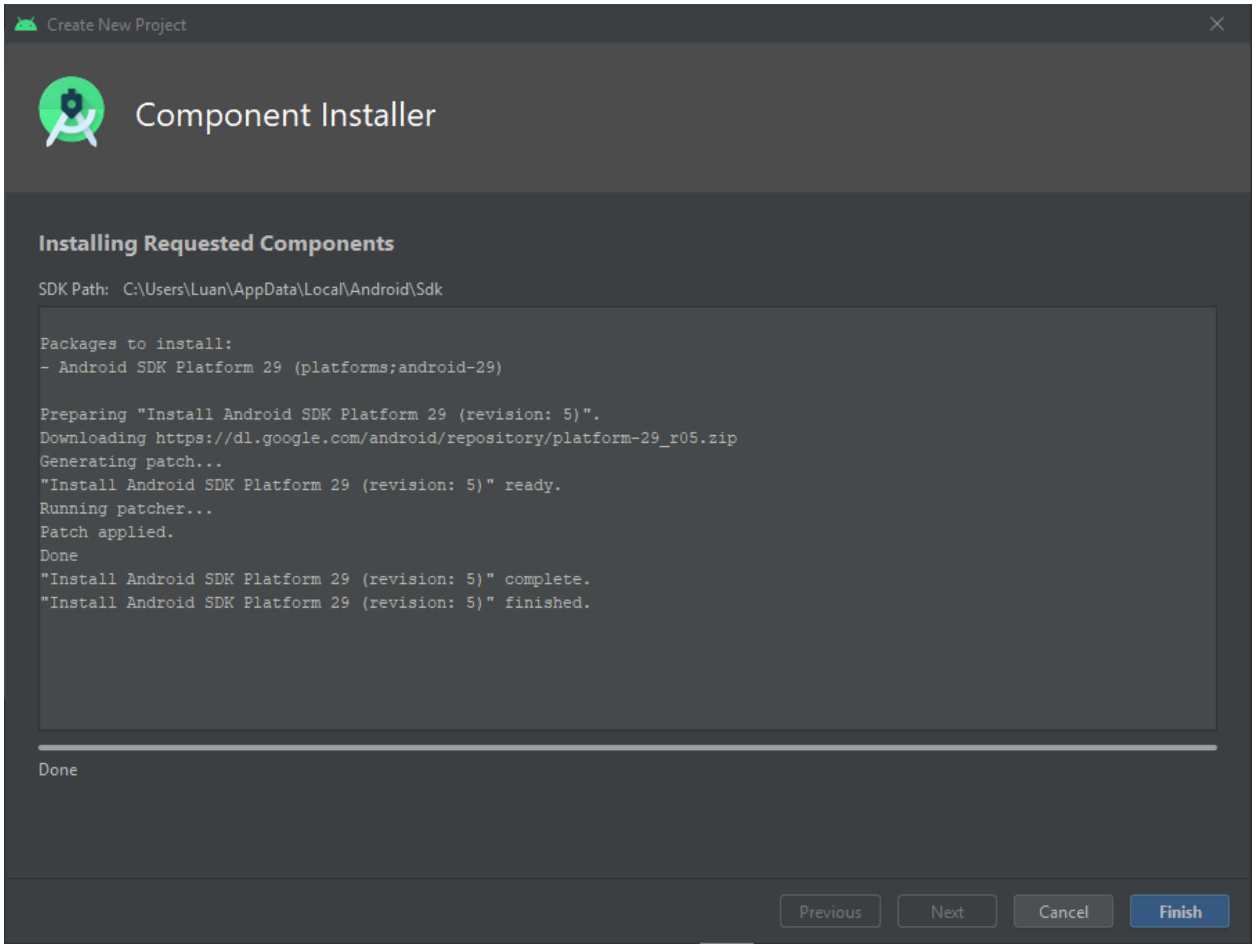
Your app will run on < 1% of devices.
Help me choose

Use legacy android.support libraries

Empty Activity

Creates a new empty activity.

4. You might have to install some components if this is your first time creating a project on Android Studio. This might take a few minutes to complete depending on your internet connection.



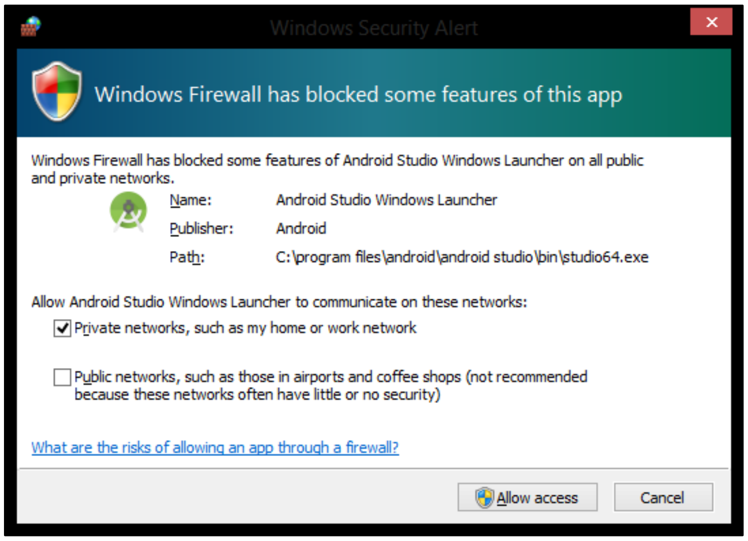
Installing Requested Components

SDK Path: C:\Users\Luan\AppData\Local\Android\Sdk

```
Packages to install:
- Android SDK Platform 29 (platforms;android-29)

Preparing "Install Android SDK Platform 29 (revision: 5)".
Downloading https://dl.google.com/android/repository/platform-29_r05.zip
Generating patch...
"Install Android SDK Platform 29 (revision: 5)" ready.
Running patcher...
Patch applied.
Done
"Install Android SDK Platform 29 (revision: 5)" complete.
"Install Android SDK Platform 29 (revision: 5)" finished.
```

Done

Note: If you are on Windows, you may get a security warning. Allow Android Studio access to continue.



Windows Security Alert

Windows Firewall has blocked some features of this app

Windows Firewall has blocked some features of Android Studio Windows Launcher on all public and private networks.

Name:	Android Studio Windows Launcher
Publisher:	Android
Path:	C:\program files\android\android studio\bin\studio64.exe

Allow Android Studio Windows Launcher to communicate on these networks:

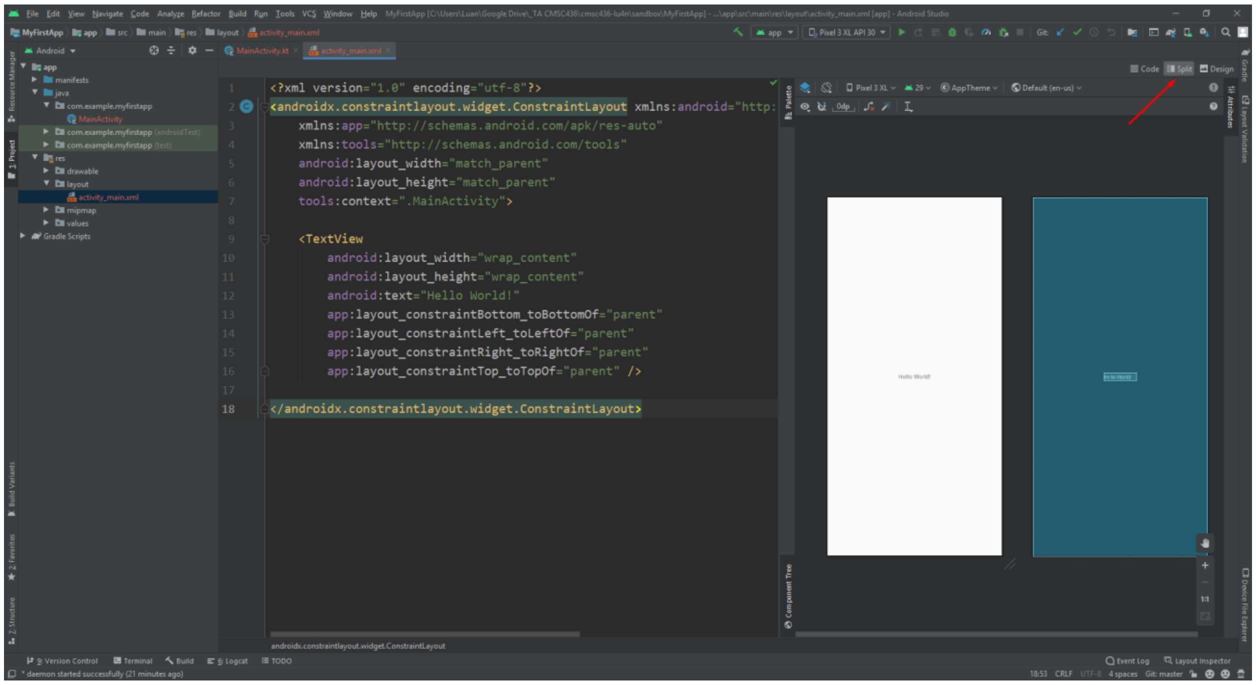
[✓] Private networks, such as my home or work network

[ ] Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)

What are the risks of allowing an app through a firewall?

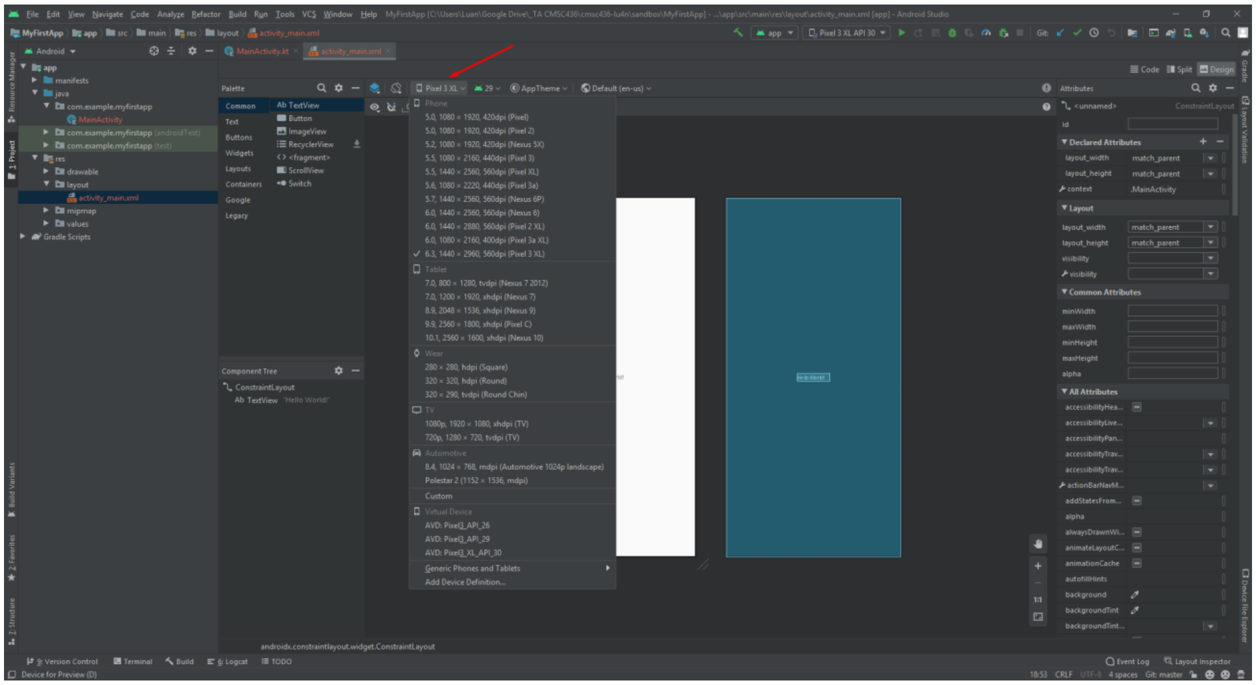
5. It may take a few minutes for the project to build depending on your computer speed. When it is completely loaded, navigate and open ‘java/com.example.myfirstapp/MainActivity.kt’ and ‘res/layout/activity_main.xml’.

   Open ‘activity_main.xml’ and on the top right corner, change the view to from ‘Design’ to ‘Split’. Now you should see the code and the design side-by-side.



```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http:
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

   To ensure that your preview looks like ours, make sure you set your preview to be ‘Pixel 3 XL’.

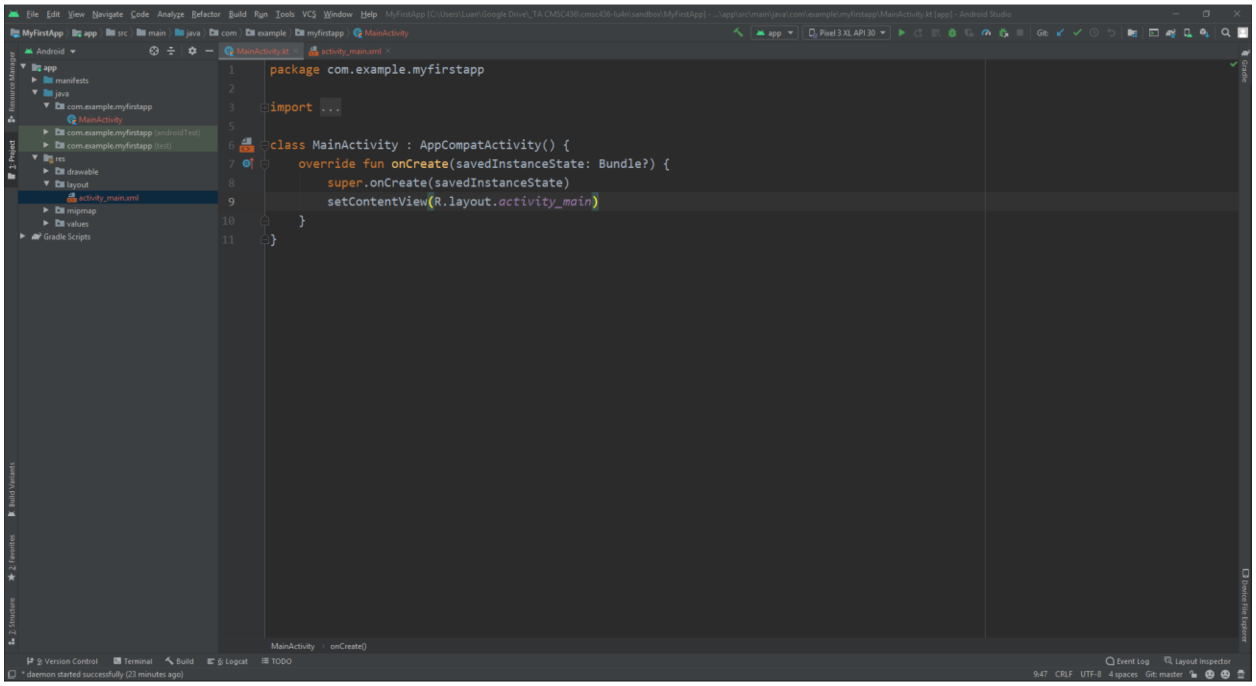

Pixel 3 XL
29
AppTheme
Default (en-us)
Phone
5.0, 1080 × 1920, 420dpi (Pixel)
5.0, 1080 × 1920, 420dpi (Pixel 2)
5.2, 1080 × 1920, 420dpi (Nexus 5X)
5.5, 1080 × 2160, 440dpi (Pixel 3)
5.5, 1440 × 2560, 560dpi (Pixel XL)
5.6, 1080 × 2220, 440dpi (Pixel 3a)
5.7, 1440 × 2560, 560dpi (Nexus 6P)
6.0, 1440 × 2560, 560dpi (Nexus 6)
6.0, 1440 × 2880, 560dpi (Pixel 2 XL)
6.0, 1080 × 2160, 400dpi (Pixel 3a XL)
✓ 6.3, 1440 × 2960, 560dpi (Pixel 3 XL)
Tablet
7.0, 800 × 1280, tvdpi (Nexus 7 2012)
7.0, 1200 × 1920, xhdpi (Nexus 7)
8.9, 2048 × 1536, xhdpi (Nexus 9)
9.9, 2560 × 1800, xhdpi (Pixel C)
10.1, 2560 × 1600, xhdpi (Nexus 10)
Wear
280 × 280, hdpi (Square)
320 × 320, hdpi (Round)
320 × 290, tvdpi (Round Chin)
TV
1080p, 1920 × 1080, xhdpi (TV)
720p, 1280 × 720, tvdpi (TV)
Automotive
8.4, 1024 × 768, mdpi (Automotive 1024p landscape)
Polestar 2 (1152 × 1536, mdpi)
Custom
Virtual Device
AVD: Pixel3_API_26
AVD: Pixel3_API_29
AVD: Pixel3_XL_API_30
Generic Phones and Tablets
Add Device Definition...

6. To view the backing code for this activity, open ‘MainActivity.kt’.

   In Kotlin, you must set a xml file that describes the layout of your app. We will be using the ‘activity_main.xml’ to describe what the layout will look like.

   When the project is built, Android will create an R object that contains all the files in the ‘res’ directory.

   In this case, we will be accessing the layout file ‘activity_main’, so we use the syntax ‘R.layout.activity_main’.



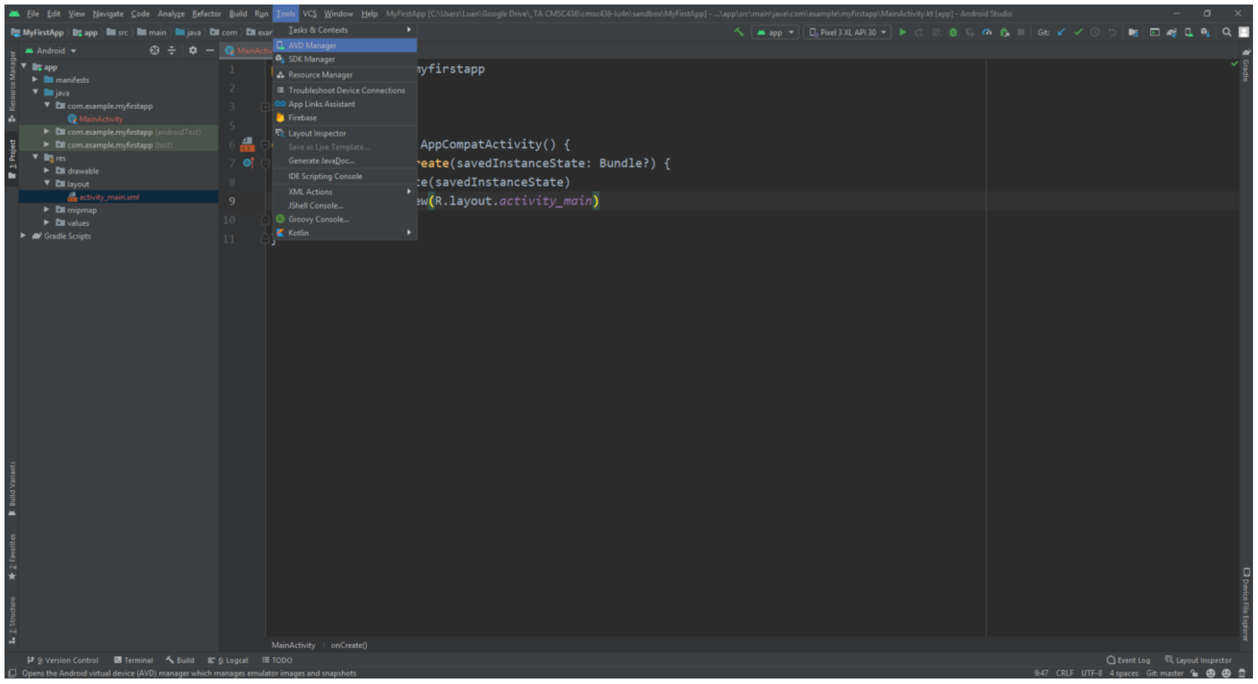
```kotlin
package com.example.myfirstapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

In Part 3, we will show you how to run the app in the Android Emulator.

Part 3 – Setting up the Emulator

In this part, we will be setting up the Android Emulator to run your project. Emulators are what Android Studio uses to simulate the actual piece of hardware your app will run on without the need for a physical Android device.

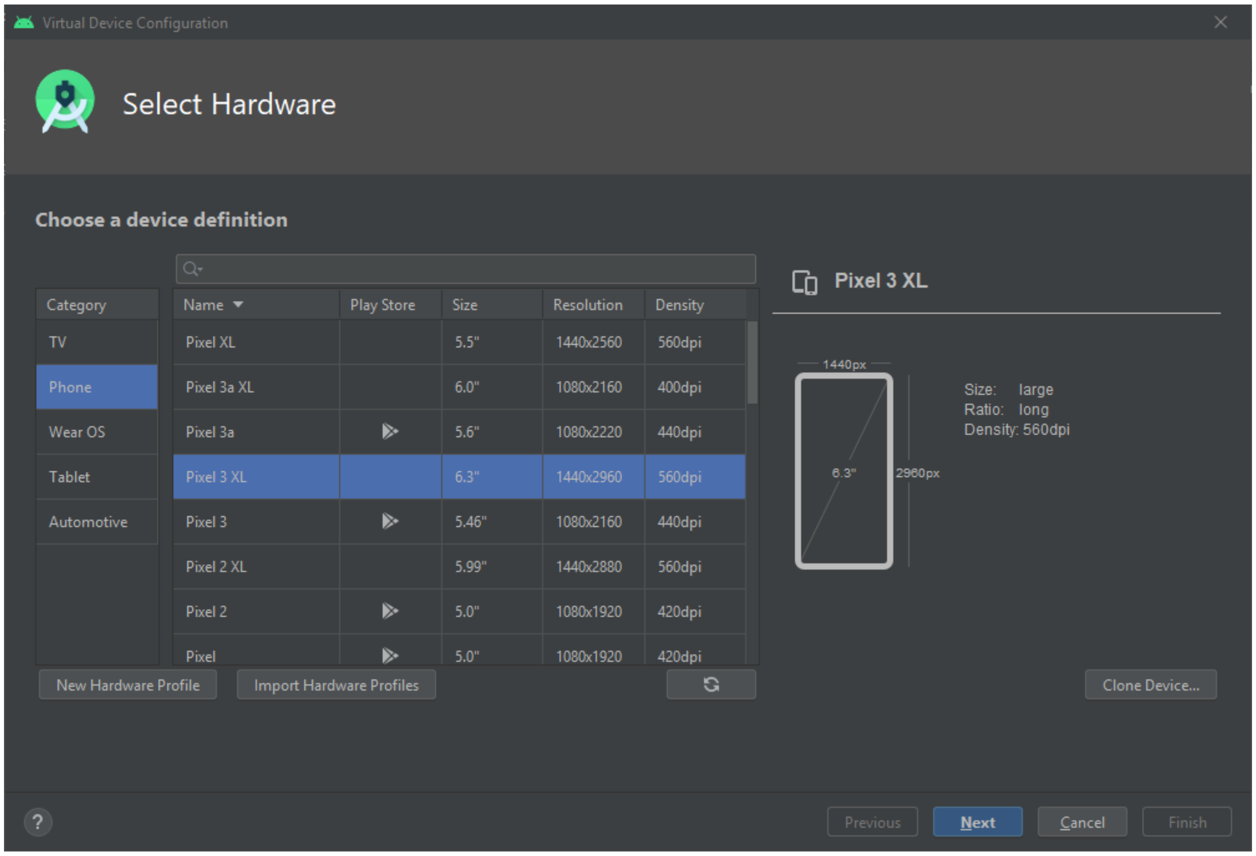
1. Open the 'AVD Manager' in the 'Tools' menu.



```
yfirstapp

 AppCompatActivity() {
reate(savedInstanceState: Bundle?) {
e(savedInstanceState)
w(R.layout.activity_main)
```

2. A new dialog box will pop up. Click ‘Create Virtual Device…’.
   Note: This step is not pictured here

   From the list of devices, select the ‘Pixel 3 XL’.



Virtual Device Configuration

Select Hardware

Choose a device definition

Category
TV
Phone
Wear OS
Tablet
Automotive

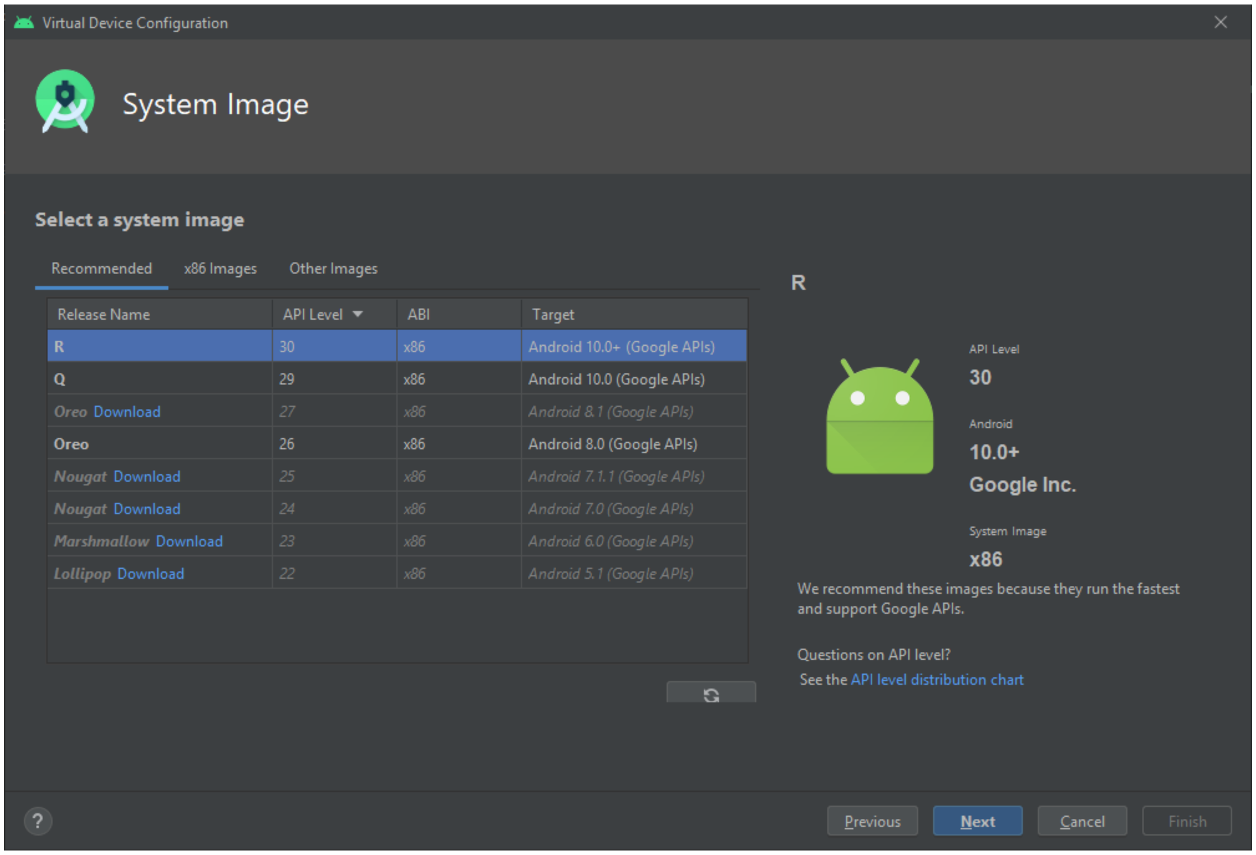
Name	Play Store	Size	Resolution	Density
Pixel XL		5.5"	1440x2560	560dpi
Pixel 3a XL		6.0"	1080x2160	400dpi
Pixel 3a	▶	5.6"	1080x2220	440dpi
Pixel 3 XL		6.3"	1440x2960	560dpi
Pixel 3	▶	5.46"	1080x2160	440dpi
Pixel 2 XL		5.99"	1440x2880	560dpi
Pixel 2	▶	5.0"	1080x1920	420dpi
Pixel	▶	5.0"	1080x1920	420dpi

Pixel 3 XL

Size: large
Ratio: long
Density: 560dpi

3. Here is list of various System Images. Each one corresponds to the Android version.

   We will be testing your code against API Level 30 (i.e. Android 10.0+ or Android R).
   Note: In this screenshot, I have Android R downloaded. However, you may have to download it on your system. This may take some time depending on your internet connection.



Select a system image

Release Name	API Level	ABI	Target
R	30	x86	Android 10.0+ (Google APIs)
Q	29	x86	Android 10.0 (Google APIs)
Oreo Download	27	x86	Android 8.1 (Google APIs)
Oreo	26	x86	Android 8.0 (Google APIs)
Nougat Download	25	x86	Android 7.1.1 (Google APIs)
Nougat Download	24	x86	Android 7.0 (Google APIs)
Marshmallow Download	23	x86	Android 6.0 (Google APIs)
Lollipop Download	22	x86	Android 5.1 (Google APIs)

R

API Level
30

Android
10.0+
Google Inc.

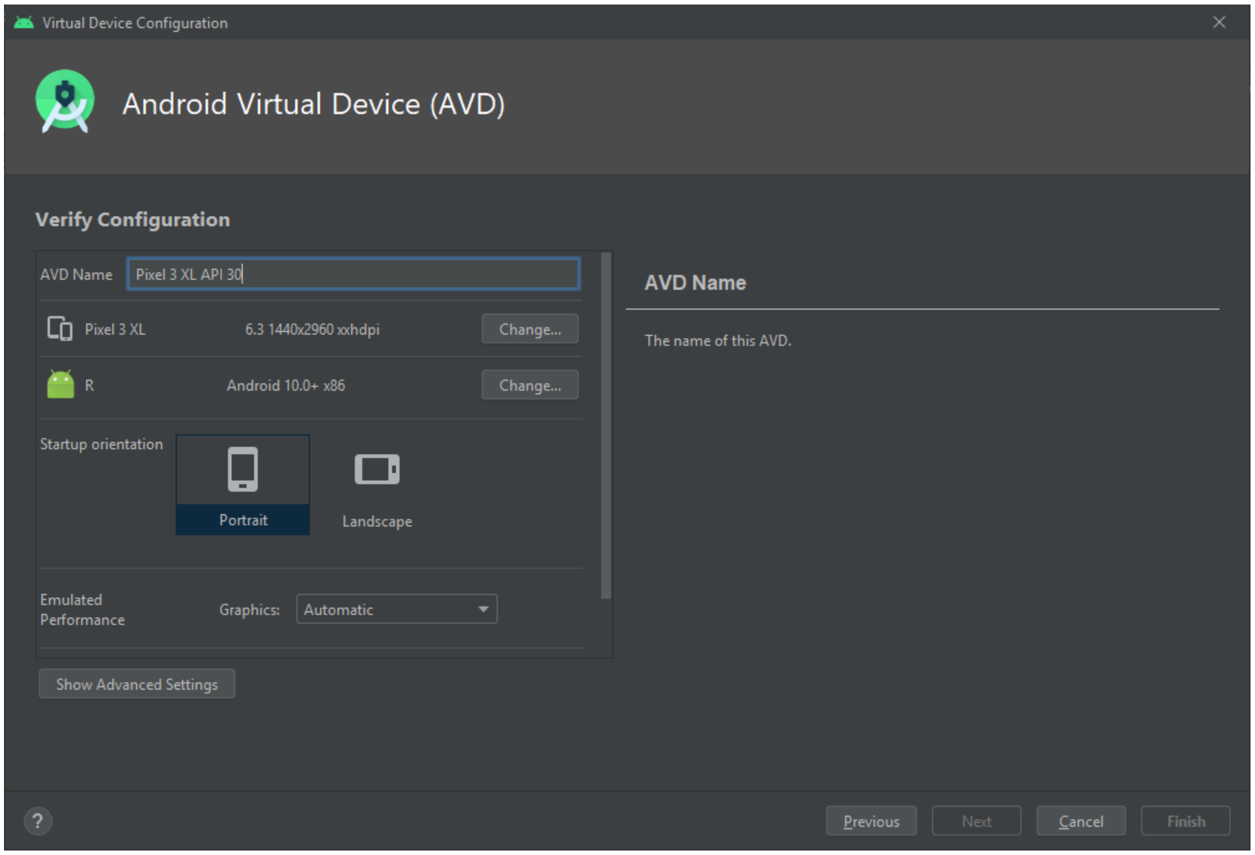
System Image
x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

4. You can rename your AVD, but we suggest just leaving it alone.

   Make sure your settings are correct and that 'Portrait' is selected.



Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

AVD Name: Pixel 3 XL API 30

Pixel 3 XL — 6.3 1440x2960 xxhdpi

R — Android 10.0+ x86

Startup orientation: Portrait

Emulated Performance — Graphics: Automatic

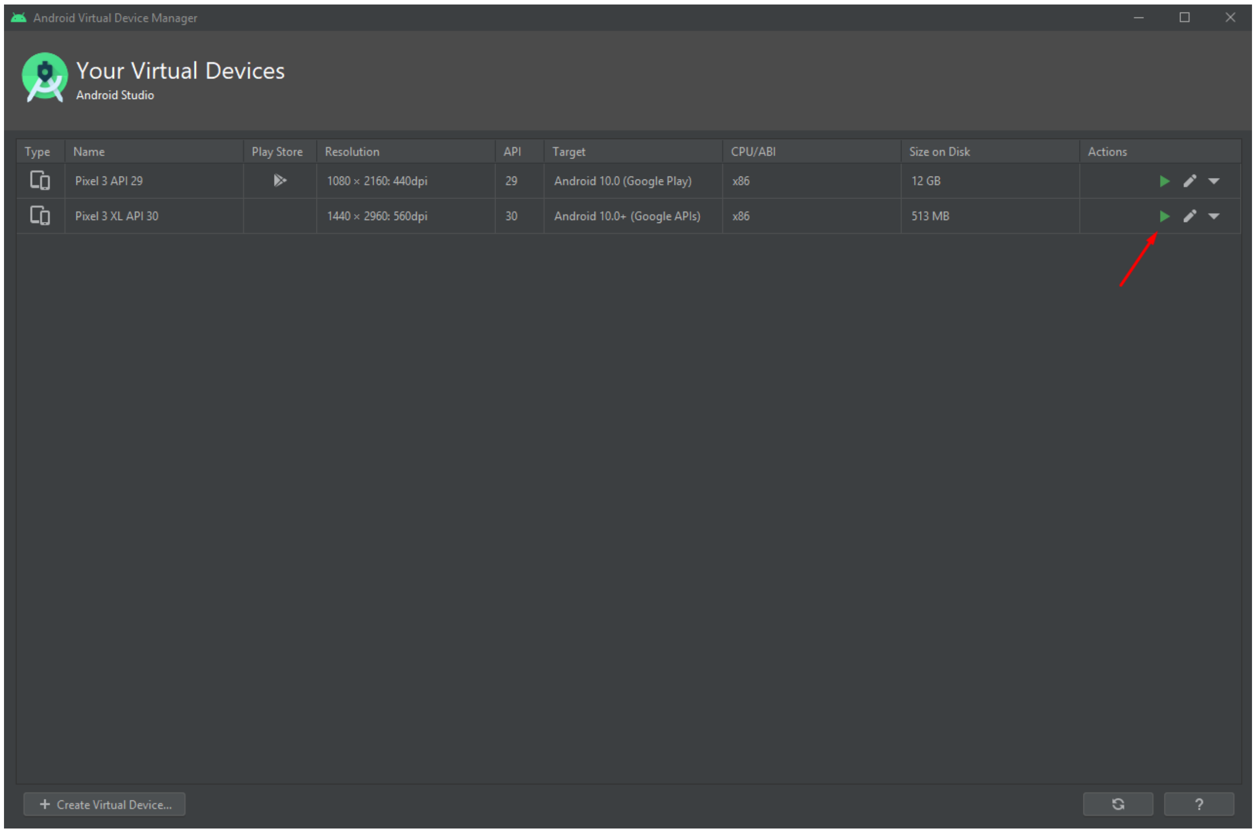
AVD Name

The name of this AVD.

5. Click the ‘Play’ icon to start the emulator.

   You can close this window once the emulator loads.

   Note: I have another AVD installed. However, you should only have one.



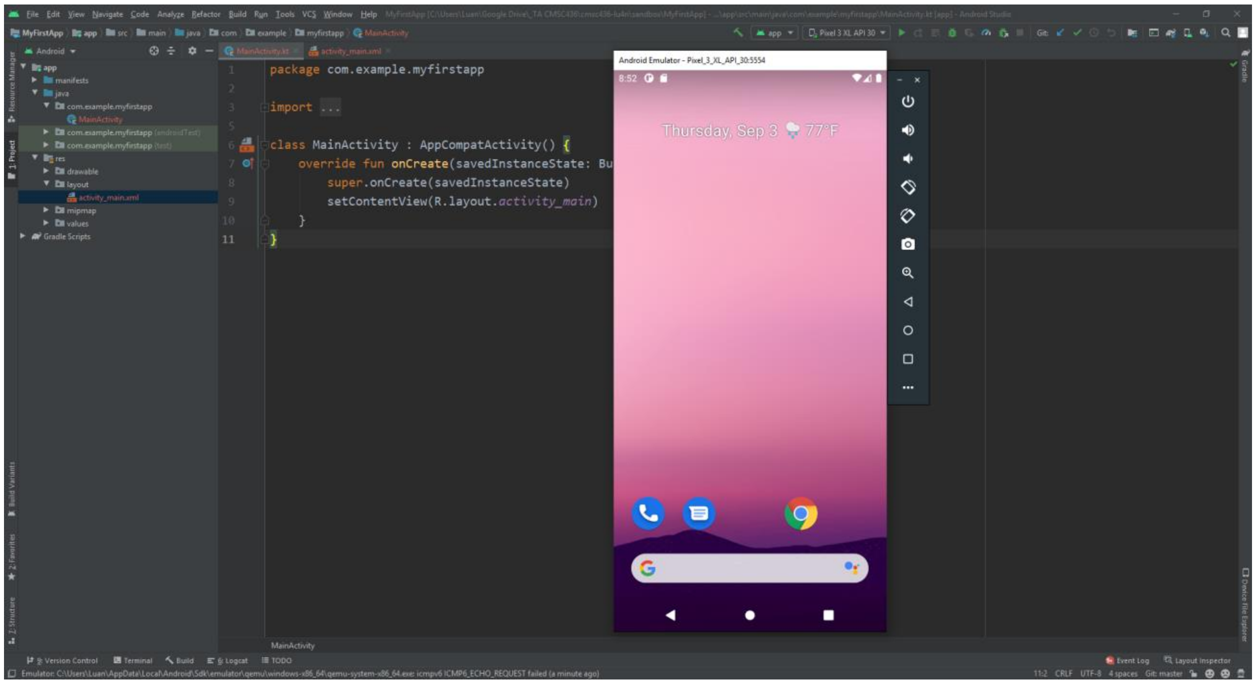
Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 3 API 29	▶	1080 × 2160: 440dpi	29	Android 10.0 (Google Play)	x86	12 GB	▶ ✎ ▾
	Pixel 3 XL API 30		1440 × 2960: 560dpi	30	Android 10.0+ (Google APIs)	x86	513 MB	▶ ✎ ▾

6. The emulator will start its boot sequence.

   Note: I have the frame turned off on my system, but you should see a Pixel 3 XL frame around yours.



```kotlin
package com.example.myfirstapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bu
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

```
Emulator: C:\Users\Luan\AppData\Local\Android\Sdk\emulator\qemu\windows-x86_64\qemu-system-x86_64.exe: icmpv6 ICMP6_ECHO_REQUEST failed (a minute ago)
```

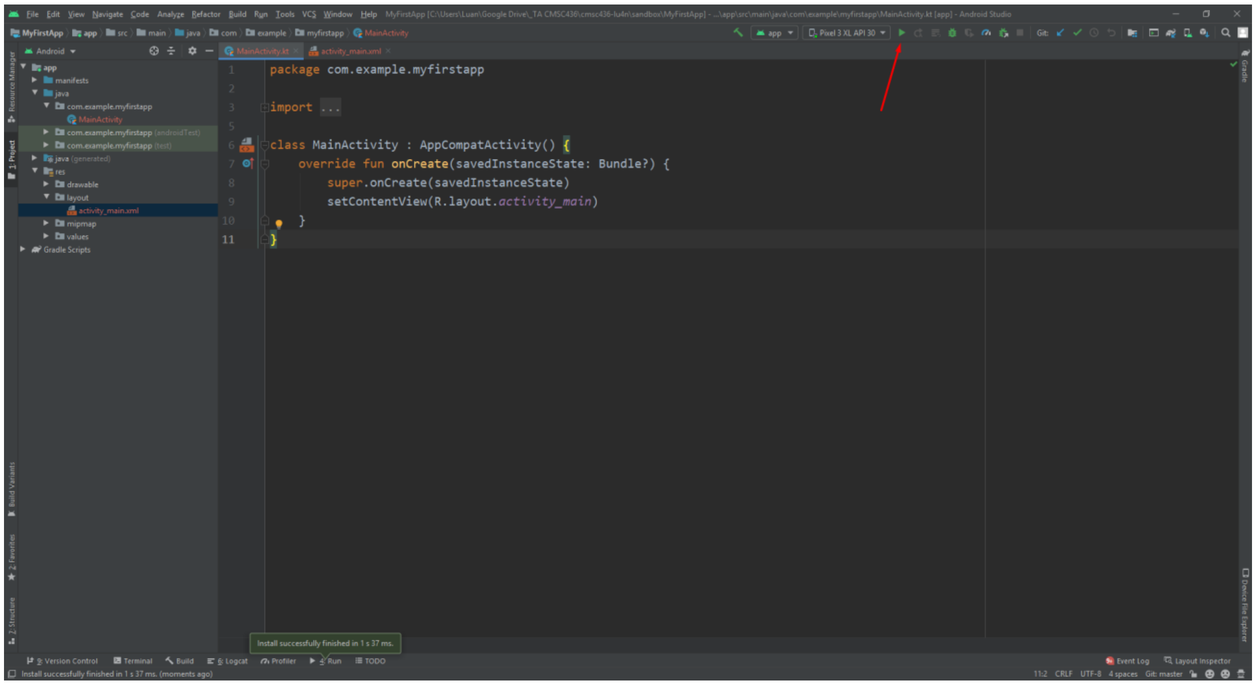
Your emulator should work just like a physical Android phone. Feel free to play around with it.

Part 4 – Running Your First App

In this part, you will learn how to run the application you created in Part 2 in the Android Emulator you created in Part 3.

There are two ways to run an app:

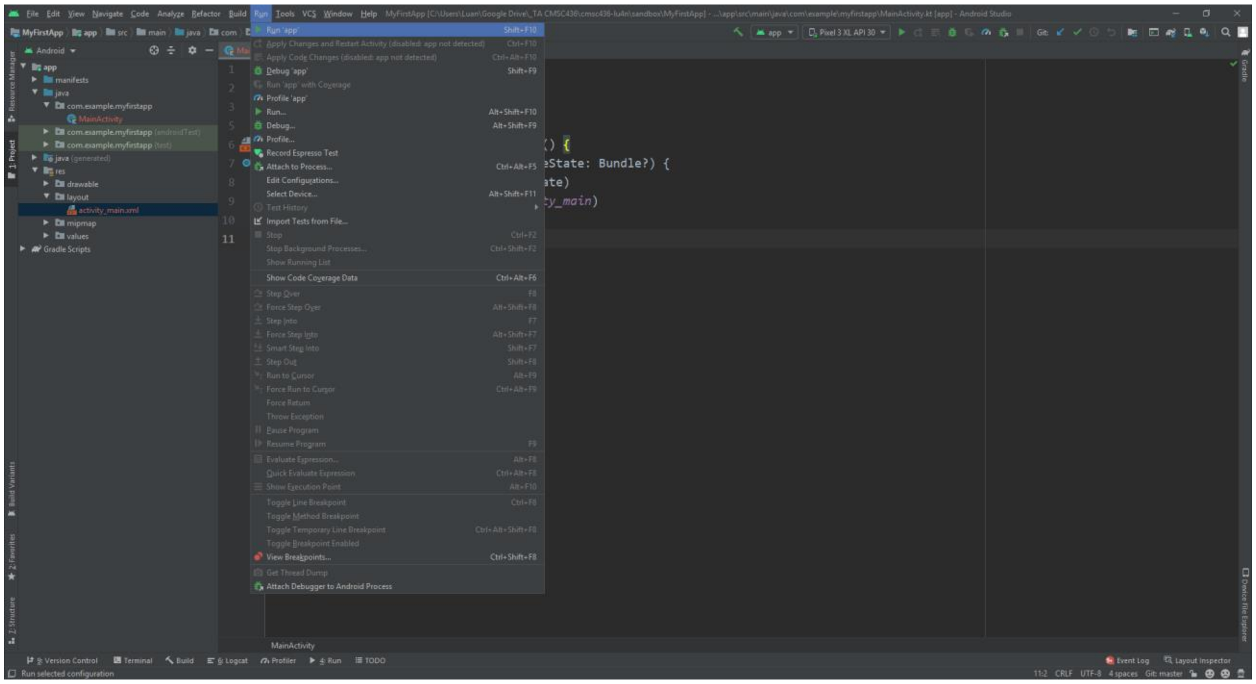
Method 1: Clicking the "Run" icon on the top bar



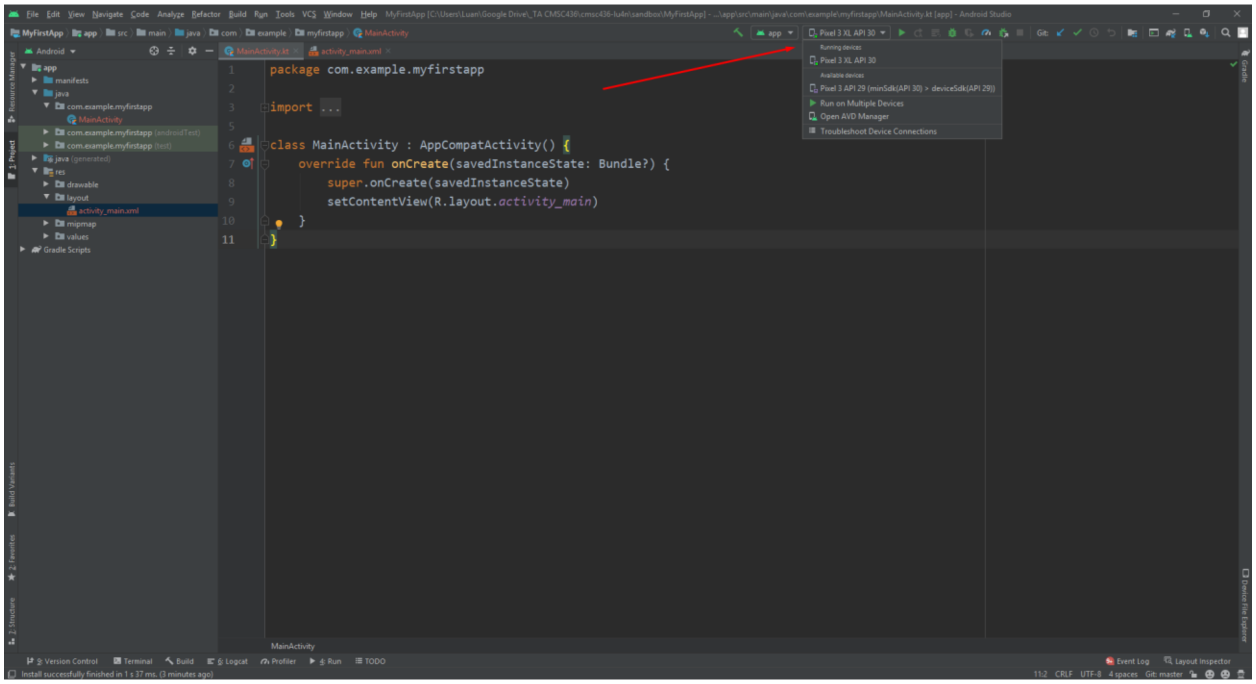
```
package com.example.myfirstapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

```
Install successfully finished in 1 s 37 ms.
Install successfully finished in 1 s 37 ms. (moments ago)
```

Method 2: selecting ‘Run app’ from the ‘Run’ menu (SHIFT + F10 on Windows).


Run
Run 'app' Shift+F10
Apply Changes and Restart Activity (disabled: app not detected) Ctrl+F10
Apply Code Changes (disabled: app not detected) Ctrl+Alt+F10
Debug 'app' Shift+F9
Run 'app' with Coverage
Profile 'app'
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Profile...
Record Espresso Test
Attach to Process... Ctrl+Alt+F5
Edit Configurations...
Select Device... Alt+Shift+F11
Test History
Import Tests from File...
Stop Ctrl+F2
Stop Background Processes... Ctrl+Shift+F2
Show Running List
Show Code Coverage Data Ctrl+Alt+F6
Step Over F8
Force Step Over Alt+Shift+F8
Step Into F7
Force Step Into Alt+Shift+F7
Smart Step Into Shift+F7
Step Out Shift+F8
Run to Cursor Alt+F9
Force Run to Cursor Ctrl+Alt+F9
Force Return
Throw Exception
Pause Program
Resume Program F9
Evaluate Expression... Alt+F8
Quick Evaluate Expression Ctrl+Alt+F8
Show Execution Point Alt+F10
Toggle Line Breakpoint Ctrl+F8
Toggle Method Breakpoint
Toggle Temporary Line Breakpoint Ctrl+Alt+Shift+F8
Toggle Breakpoint Enabled
View Breakpoints... Ctrl+Shift+F8
Get Thread Dump
Attach Debugger to Android Process

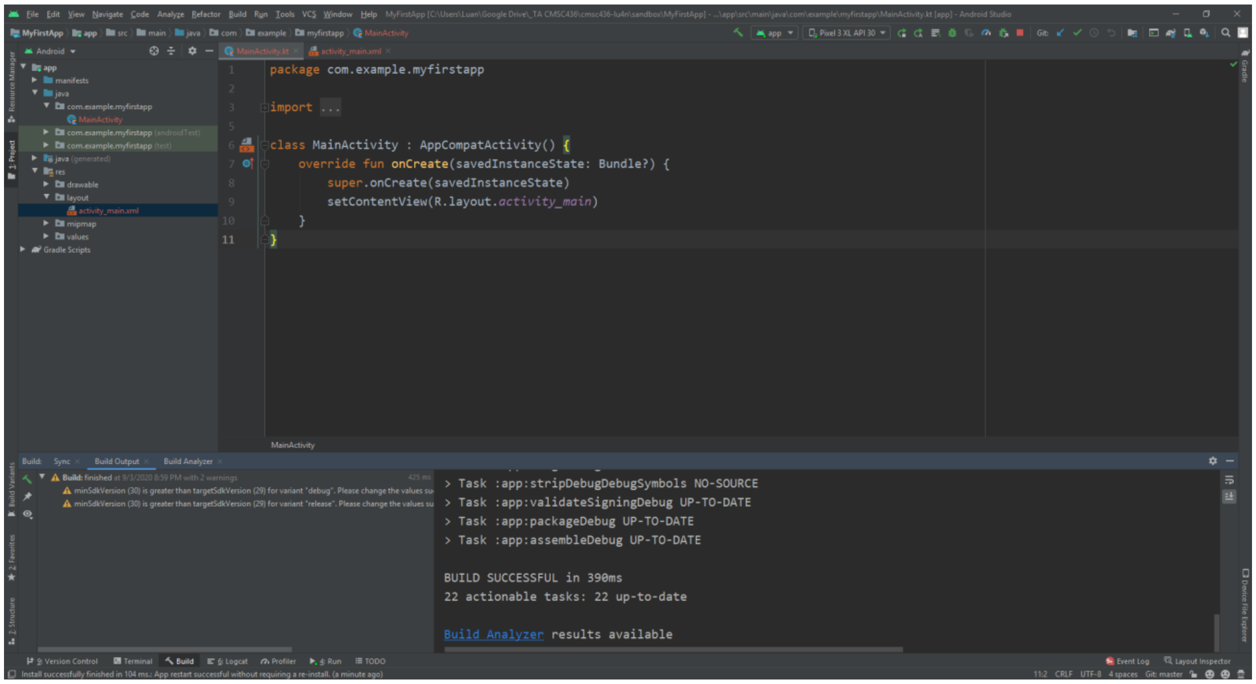
If it does not run, click on the device selection menu and select the ‘Pixel 3 XL API 30’ you created in Part 3.



```kotlin
package com.example.myfirstapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Install successfully finished in 1 s 37 ms. (3 minutes ago)

The Build Console, below the editor will show you the app being built, loaded, and configured.



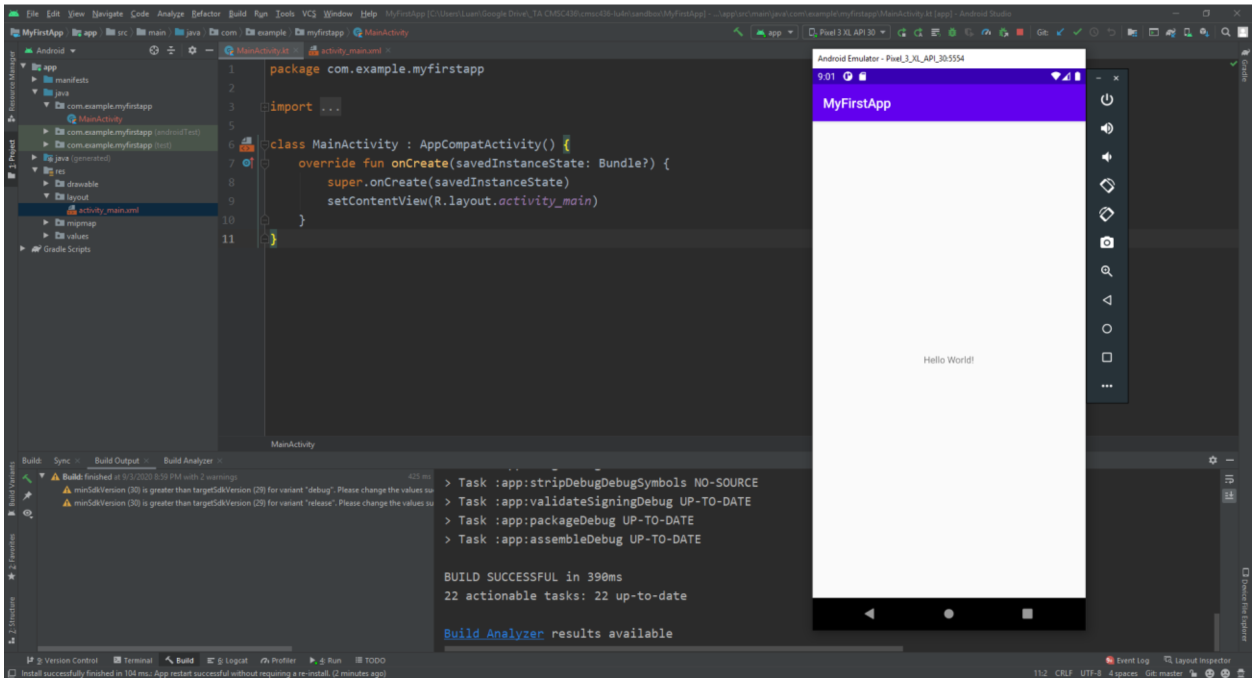
```kotlin
package com.example.myfirstapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

```
Build: finished at 9/3/2020 8:59 PM with 2 warnings    425 ms
  minSdkVersion (30) is greater than targetSdkVersion (29) for variant "debug". Please change the values su
  minSdkVersion (30) is greater than targetSdkVersion (29) for variant "release". Please change the values su
```

```
> Task :app:stripDebugDebugSymbols NO-SOURCE
> Task :app:validateSigningDebug UP-TO-DATE
> Task :app:packageDebug UP-TO-DATE
> Task :app:assembleDebug UP-TO-DATE

BUILD SUCCESSFUL in 390ms
22 actionable tasks: 22 up-to-date

Build Analyzer results available
```

Install successfully finished in 104 ms.: App restart successful without requiring a re-install. (a minute ago)

Return to your Emulator and you should see your app running.



```
package com.example.myfirstapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

```
Build: finished at 9/3/2020 8:59 PM with 2 warnings    425 ms
    minSdkVersion (30) is greater than targetSdkVersion (29) for variant "debug". Please change the values su
    minSdkVersion (30) is greater than targetSdkVersion (29) for variant "release". Please change the values su
```

```
> Task :app:stripDebugDebugSymbols NO-SOURCE
> Task :app:validateSigningDebug UP-TO-DATE
> Task :app:packageDebug UP-TO-DATE
> Task :app:assembleDebug UP-TO-DATE

BUILD SUCCESSFUL in 390ms
22 actionable tasks: 22 up-to-date

Build Analyzer results available
```

```
Install successfully finished in 104 ms.: App restart successful without requiring a re-install. (2 minutes ago)
```

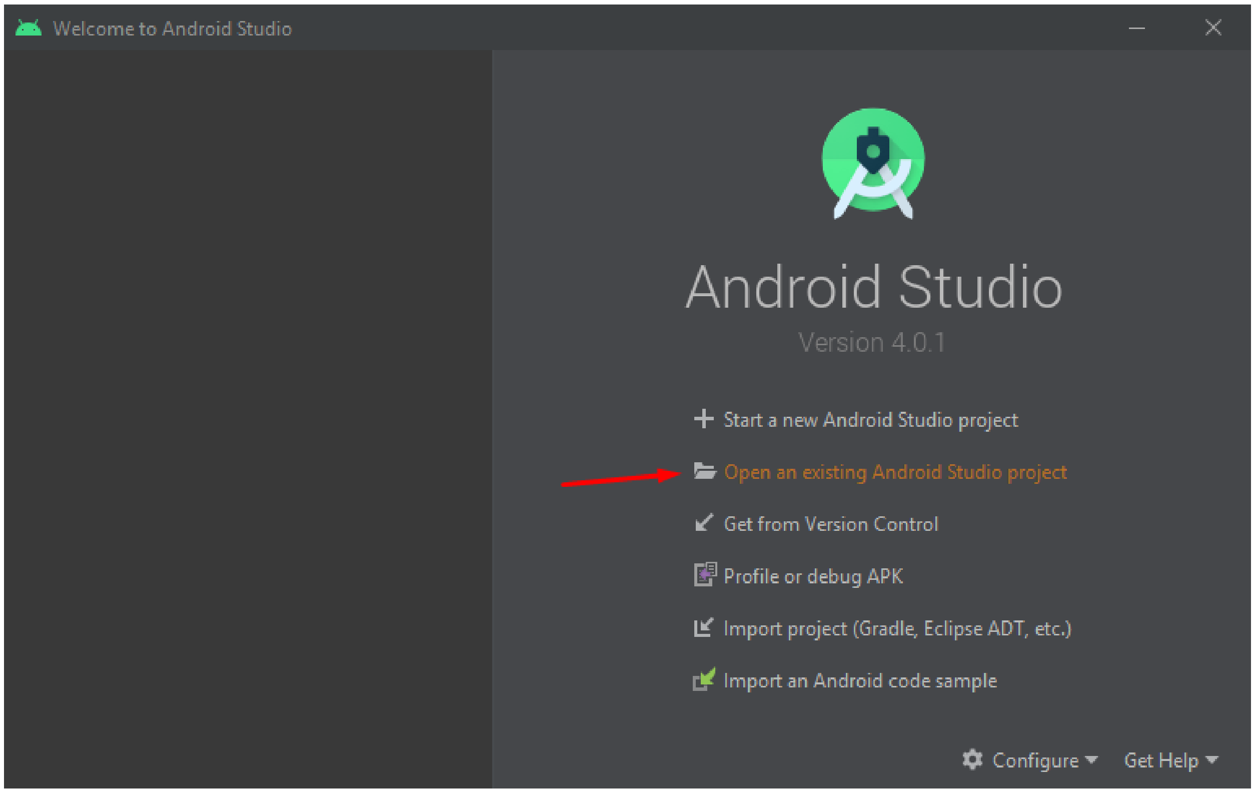
Feel free to mess around with the layout file and see how that changes the app’s layout.

Part 5 – Importing an Existing Project

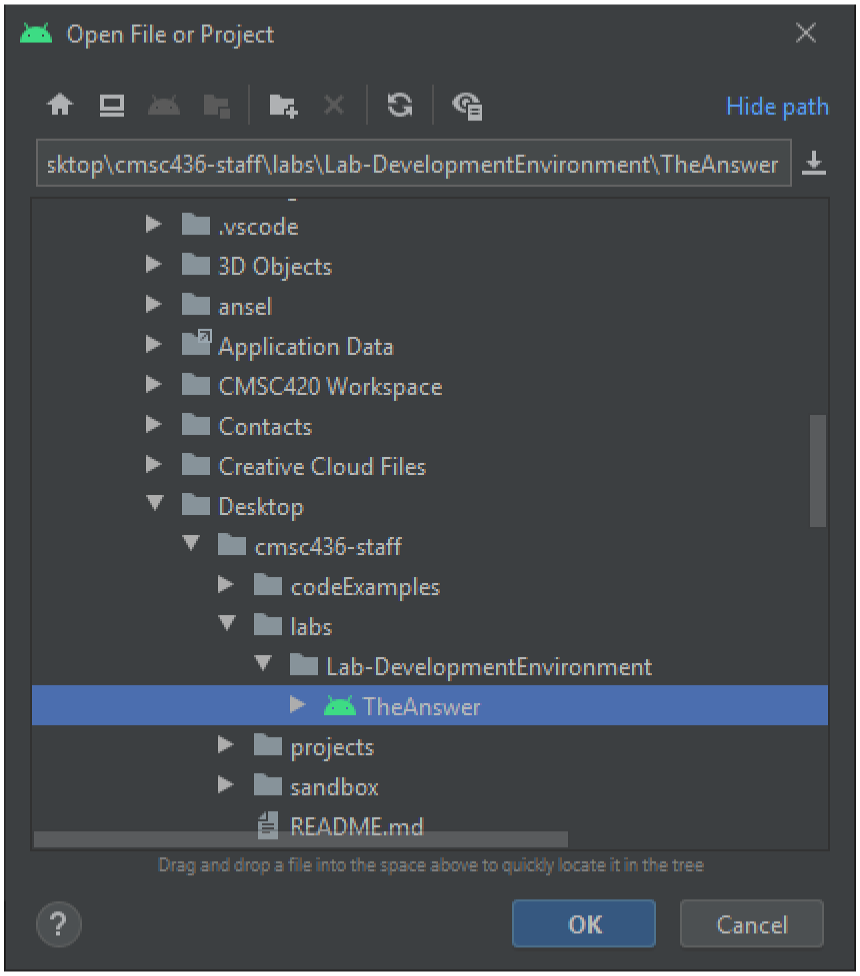
In this part, you will learn how to import a project into Android Studio. Make sure you pull from the upstream repository to get the skeleton code for the project.

Inside the Lab-DevelopmentEnvironment directory, you should see a directory named ‘TheAnswer’. This is the project we will be working with.

Open Android Studio again and select ‘Open an existing Android Studio project’.


Welcome to Android Studio
Android Studio
Version 4.0.1
Start a new Android Studio project
Open an existing Android Studio project
Get from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

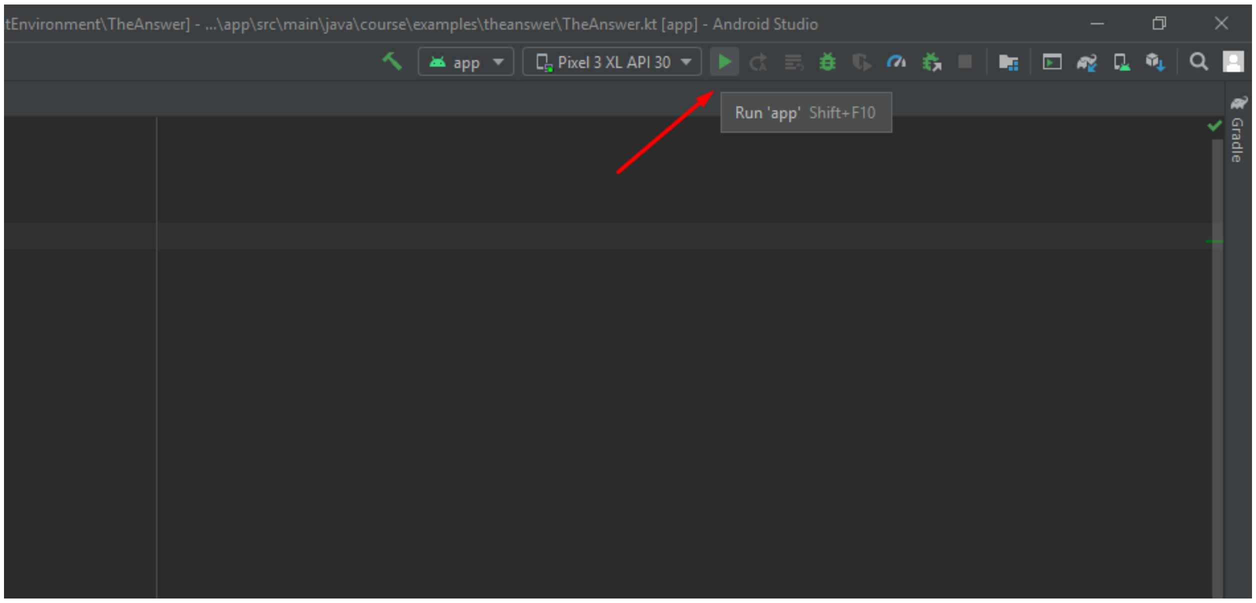
Next, you want to browse and select the project that you want to open. In this case, select 'TheAnswer' from where you cloned your repository in your local system. Then press 'OK'.

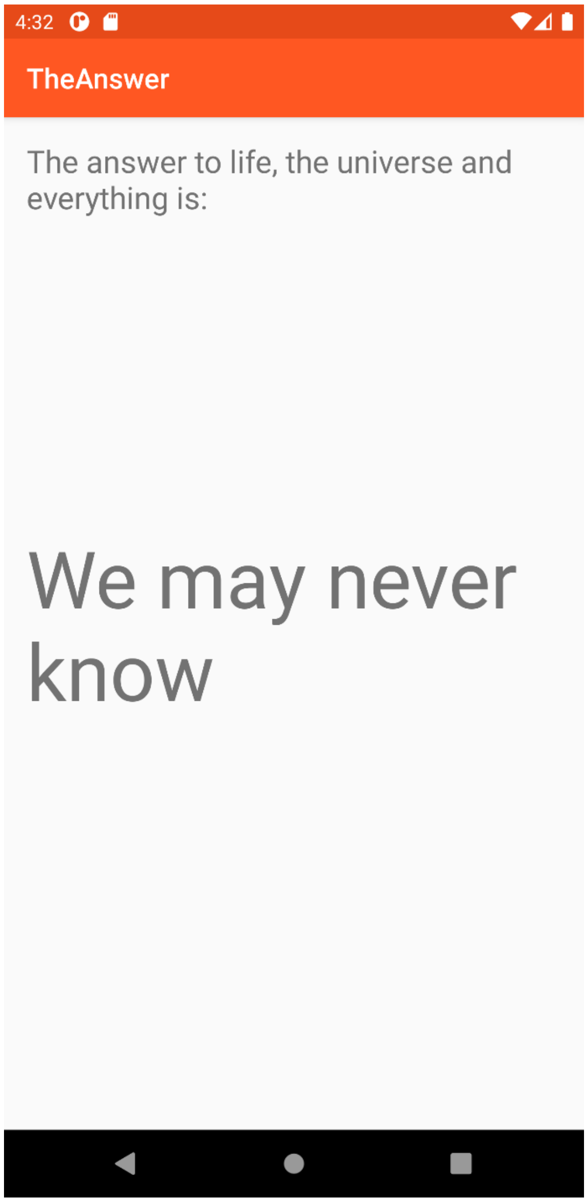

Open File or Project
Hide path
Drag and drop a file into the space above to quickly locate it in the tree
OK
Cancel

```
sktop\cmsc436-staff\labs\Lab-DevelopmentEnvironment\TheAnswer
```

- .vscode
- 3D Objects
- ansel
- Application Data
- CMSC420 Workspace
- Contacts
- Creative Cloud Files
- Desktop
  - cmsc436-staff
    - codeExamples
    - labs
      - Lab-DevelopmentEnvironment
        - TheAnswer
    - projects
    - sandbox
    - README.md

When it is done building launch the app and select the AVD you created at the start of the tutorial. The Android emulator will now run the example app.


tEnvironment\TheAnswer] - ...\app\src\main\java\course\examples\theanswer\TheAnswer.kt [app] - Android Studio
app
Pixel 3 XL API 30
Run 'app' Shift+F10
Gradle



The answer to life, the universe and everything is:

We may never know

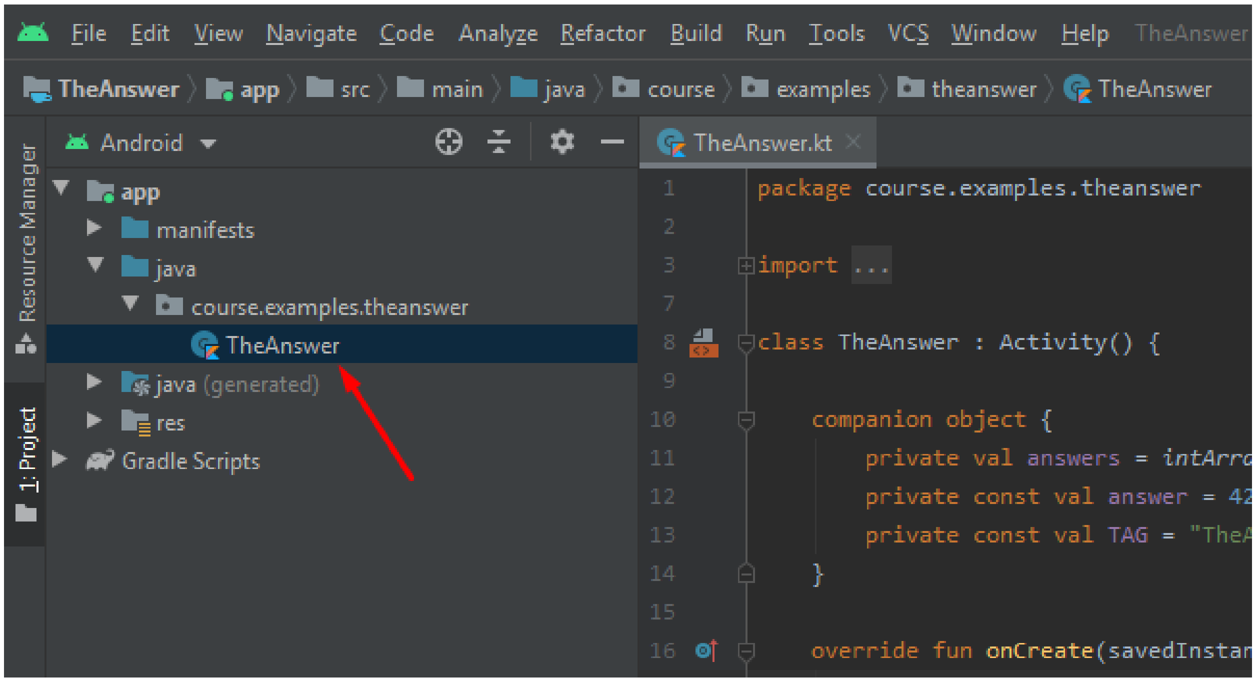
Currently, the app does not return the correct response. Look at the bottom of 'TheAnswer.kt' for the 'findAnswer()' method. Try to change the code to make it display the correct answer: 42.

Hint: you only need to change one line of code.

Part 6 – Debugging an Android App

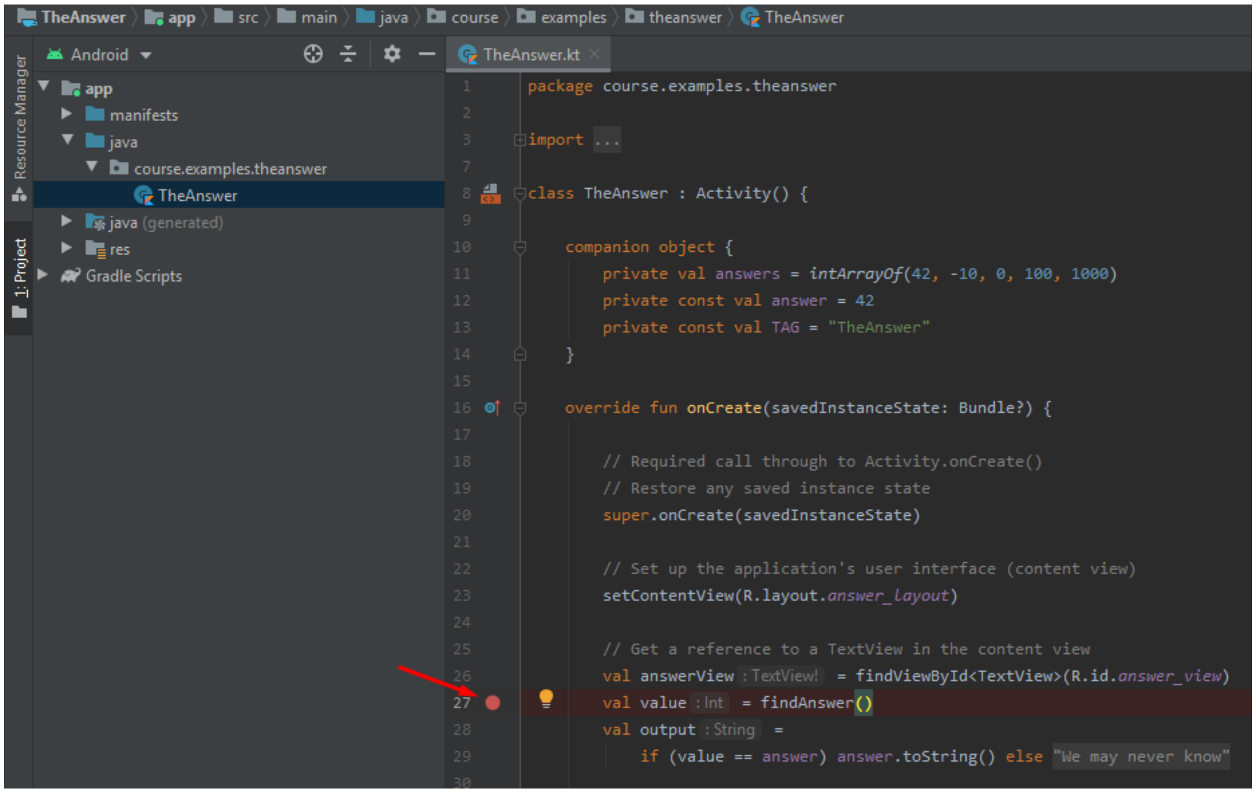
In this part, we will use the debugger to debug TheAnswer app.

1. On the left side of the editor, open 'TheAnswer.kt'. It should be under app > java > course.examples.theanswer > TheAnswer.



```
package course.examples.theanswer

import ...

class TheAnswer : Activity() {

    companion object {
        private val answers = intArra
        private const val answer = 42
        private const val TAG = "TheA
    }

    override fun onCreate(savedInstan
```

2. Click on the highlighted area next to the line: ‘val value = findAnswer()’


TheAnswer › app › src › main › java › course › examples › theanswer › TheAnswer
Android
app
manifests
java
course.examples.theanswer
TheAnswer
java (generated)
res
Gradle Scripts
Resource Manager
1: Project
TheAnswer.kt

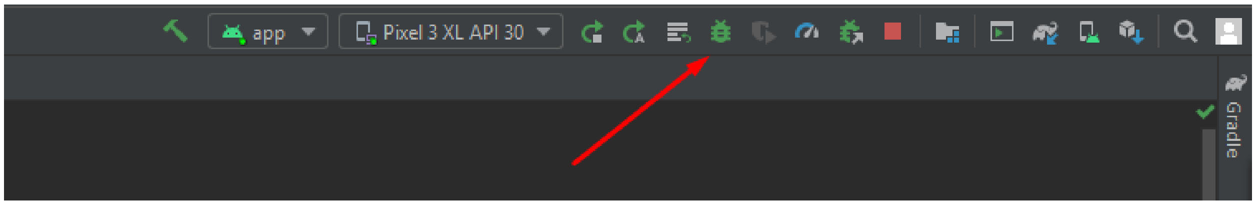
```
package course.examples.theanswer

import ...

class TheAnswer : Activity() {

    companion object {
        private val answers = intArrayOf(42, -10, 0, 100, 1000)
        private const val answer = 42
        private const val TAG = "TheAnswer"
    }

    override fun onCreate(savedInstanceState: Bundle?) {

        // Required call through to Activity.onCreate()
        // Restore any saved instance state
        super.onCreate(savedInstanceState)

        // Set up the application's user interface (content view)
        setContentView(R.layout.answer_layout)

        // Get a reference to a TextView in the content view
        val answerView : TextView! = findViewById<TextView>(R.id.answer_view)
        val value : Int = findAnswer()
        val output : String =
            if (value == answer) answer.toString() else "We may never know"
```

A new breakpoint will be set at that line.

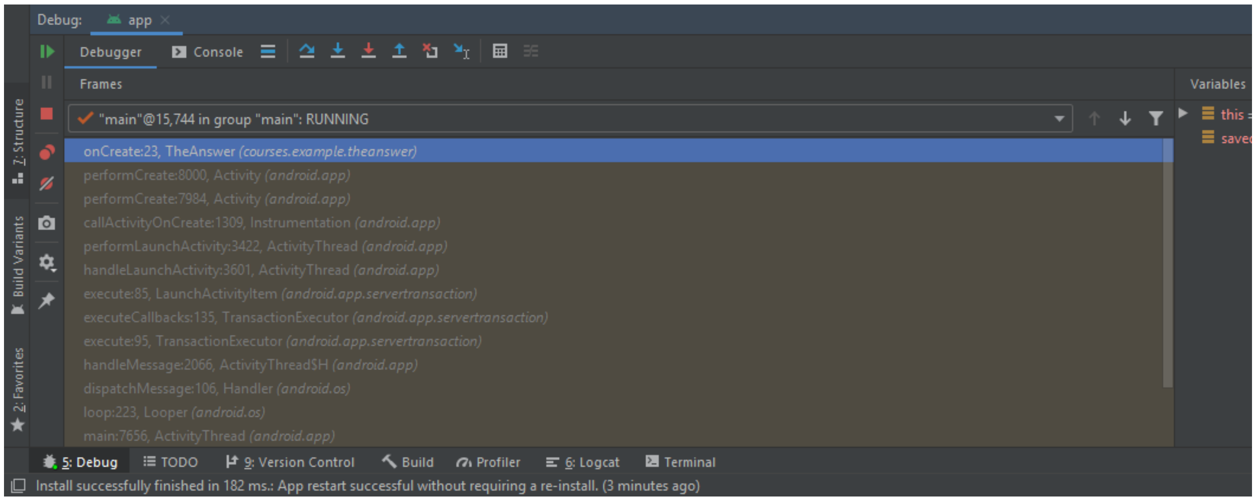
3. Instead of running the program using the ‘Play’ icon, press the ‘Debug’ icon.

   Debugging will stop the program at breakpoints that you have set.

   You can use these breakpoints to see what the app is doing from line to line.



4. If it does not automatically, switch to the ‘Debug’ pane at the bottom.



```
"main"@15,744 in group "main": RUNNING
onCreate:23, TheAnswer (courses.example.theanswer)
performCreate:8000, Activity (android.app)
performCreate:7984, Activity (android.app)
callActivityOnCreate:1309, Instrumentation (android.app)
performLaunchActivity:3422, ActivityThread (android.app)
handleLaunchActivity:3601, ActivityThread (android.app)
execute:85, LaunchActivityItem (android.app.servertransaction)
executeCallbacks:135, TransactionExecutor (android.app.servertransaction)
execute:95, TransactionExecutor (android.app.servertransaction)
handleMessage:2066, ActivityThread$H (android.app)
dispatchMessage:106, Handler (android.os)
loop:223, Looper (android.os)
main:7656, ActivityThread (android.app)
```

```
this
save
```

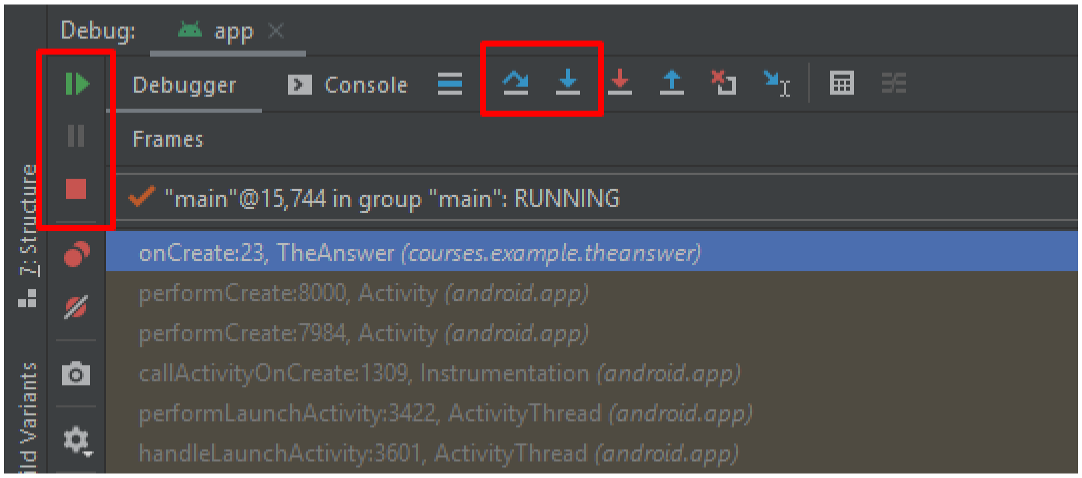
```
Install successfully finished in 182 ms.: App restart successful without requiring a re-install. (3 minutes ago)
```

5. The app should stop at line 27 and nothing should show on your AVD.

   The debugger has a few tools that we will look at.

   The top bar are the controls for executing code by line. The first one is 'Step Over' and the second is ''Step Into'. Use 'Step Over' is used to go to the next line of code. However, if the line calls a method you may want to use 'Step Into' to enter code in the method.

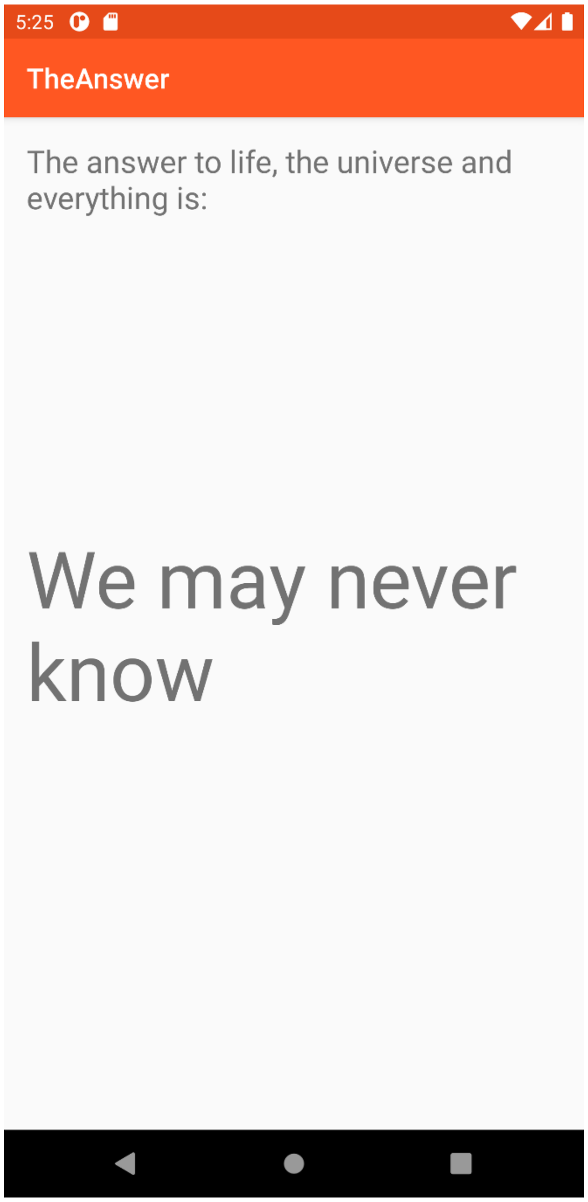
   The side bar has three controls for running the app. The top one is 'Resume Program' and the bottom one is 'Stop app'. Use 'Resume Program' to continue running the app until it hits another breakpoint. Use the 'Stop app' if you want to completely stop the app.



```
"main"@15,744 in group "main": RUNNING
onCreate:23, TheAnswer (courses.example.theanswer)
performCreate:8000, Activity (android.app)
performCreate:7984, Activity (android.app)
callActivityOnCreate:1309, Instrumentation (android.app)
performLaunchActivity:3422, ActivityThread (android.app)
handleLaunchActivity:3601, ActivityThread (android.app)
```

Play around with setting breakpoints and the debugger until you are comfortable with it.

6. Press the 'Resume Program' icon to continue executing the app. The app will finish loading and display the text again.



The answer to life, the universe and everything is:

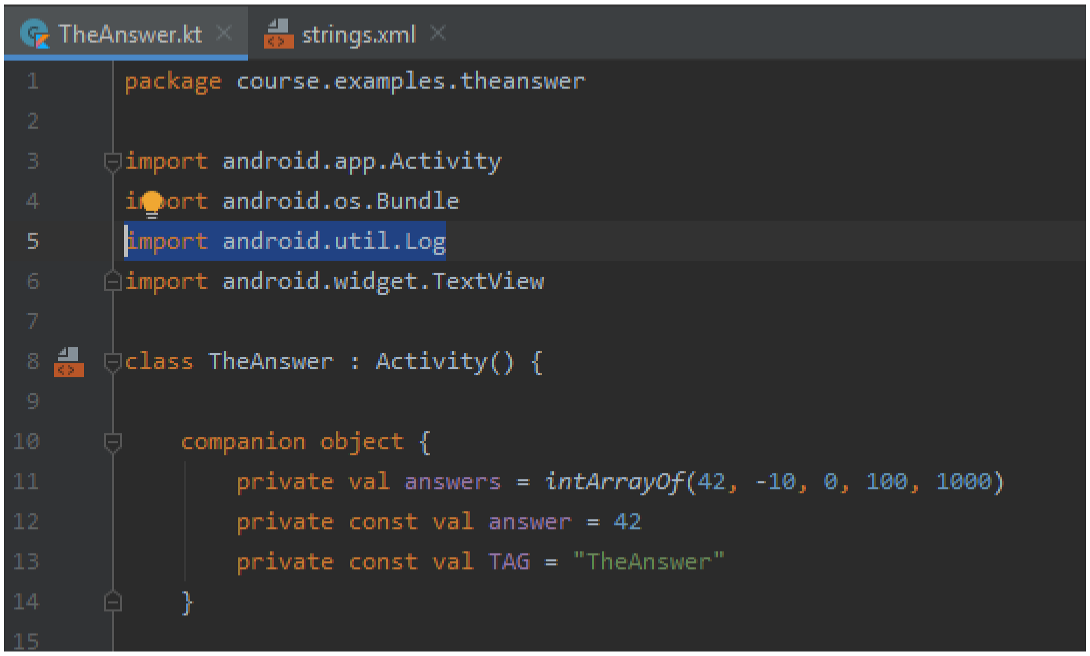
We may never know

The next debugging task will have you create and display informational messages to the Logcat panel, to help you better understand the application's runtime behavior. To generate these messages, you will use methods in the android.util.Log class. You will also need to import this class into your app. Some Logcat functions include:

1. Log.i(TAG, …) – Sends an INFO Logcat message
2. Log.d(TAG, …) – Sends a DEBUG Logcat message
3. Log.e(TAG, …) – Sends an ERROR Logcat message
4. Log.v(TAG, …) – Sends a VERBOSE Logcat message

See https://developer.android.com/reference/android/util/Log.html for more information.

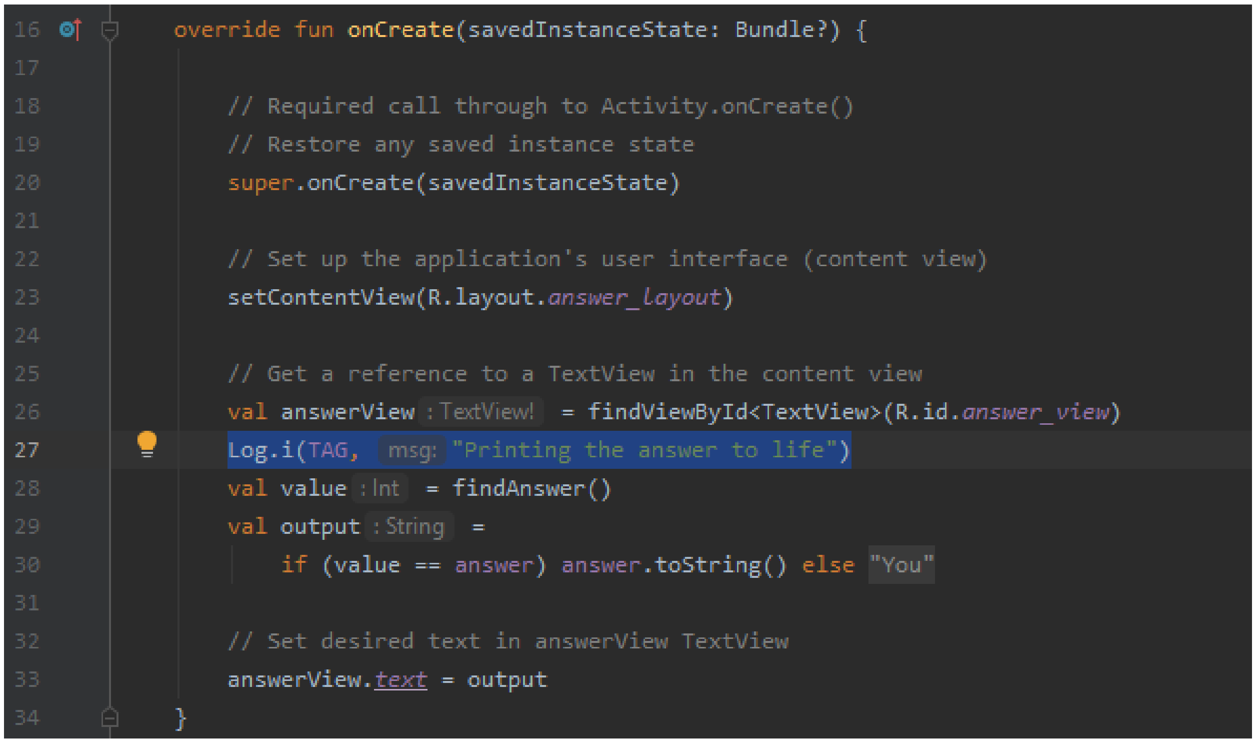
1. Import the 'android.util.Log' library by typing, "import android.util.Log" in TheAnswer.kt.



```
package course.examples.theanswer

import android.app.Activity
import android.os.Bundle
import android.util.Log
import android.widget.TextView

class TheAnswer : Activity() {

    companion object {
        private val answers = intArrayOf(42, -10, 0, 100, 1000)
        private const val answer = 42
        private const val TAG = "TheAnswer"
    }

```

The Log class' methods require a String called Tag, which identifies the creator of the message and can be used to sort and filter messages that are displayed. This is on line 13 inside the 'companion object'.

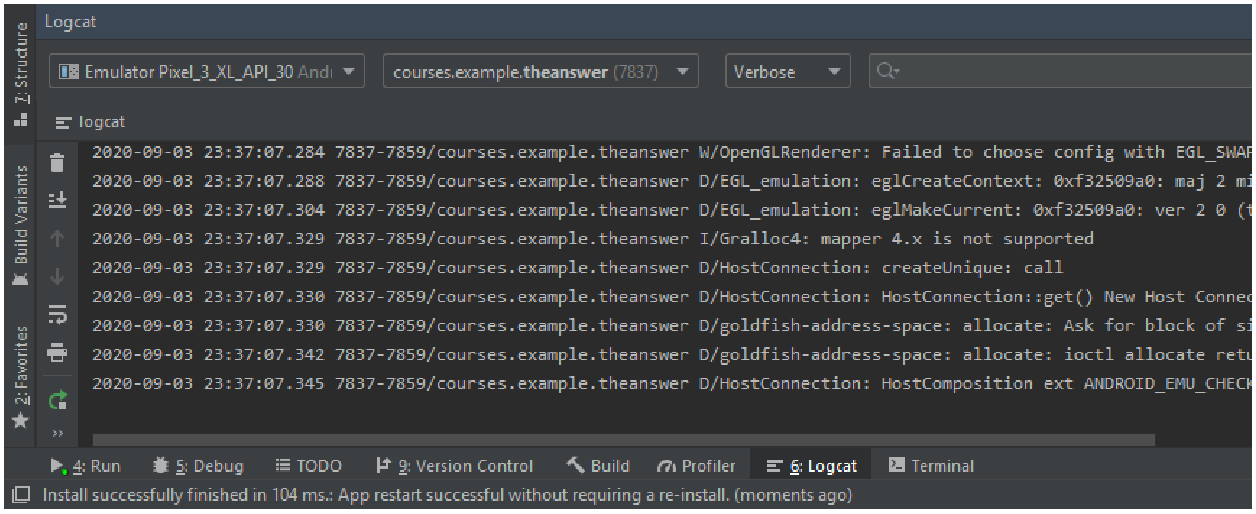
2. Use the ‘Log.i()’ function to create a new output message before the ‘val value = findAnswer()’. Type a new line: ‘Log.i(TAG, “Printing the answer to life”)’



```
    override fun onCreate(savedInstanceState: Bundle?) {

        // Required call through to Activity.onCreate()
        // Restore any saved instance state
        super.onCreate(savedInstanceState)

        // Set up the application's user interface (content view)
        setContentView(R.layout.answer_layout)

        // Get a reference to a TextView in the content view
        val answerView : TextView! = findViewById<TextView>(R.id.answer_view)
        Log.i(TAG, msg: "Printing the answer to life")
        val value : Int = findAnswer()
        val output : String =
            if (value == answer) answer.toString() else "You"

        // Set desired text in answerView TextView
        answerView.text = output
    }
```

3. Save your changes and run the application.
4. If you have not already, stop the app.

Run the app as usual and switch to the ‘Logcat’ pane at the bottom.

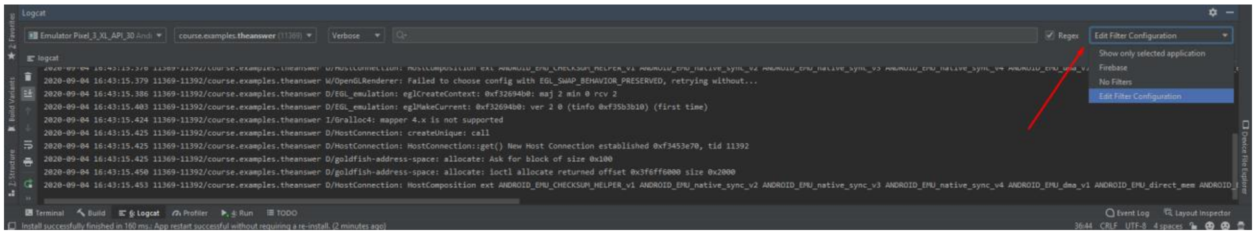


```
2020-09-03 23:37:07.284 7837-7859/courses.example.theanswer W/OpenGLRenderer: Failed to choose config with EGL_SWAP
2020-09-03 23:37:07.288 7837-7859/courses.example.theanswer D/EGL_emulation: eglCreateContext: 0xf32509a0: maj 2 mi
2020-09-03 23:37:07.304 7837-7859/courses.example.theanswer D/EGL_emulation: eglMakeCurrent: 0xf32509a0: ver 2 0 (t
2020-09-03 23:37:07.329 7837-7859/courses.example.theanswer I/Gralloc4: mapper 4.x is not supported
2020-09-03 23:37:07.329 7837-7859/courses.example.theanswer D/HostConnection: createUnique: call
2020-09-03 23:37:07.330 7837-7859/courses.example.theanswer D/HostConnection: HostConnection::get() New Host Connec
2020-09-03 23:37:07.330 7837-7859/courses.example.theanswer D/goldfish-address-space: allocate: Ask for block of si
2020-09-03 23:37:07.342 7837-7859/courses.example.theanswer D/goldfish-address-space: allocate: ioctl allocate retu
2020-09-03 23:37:07.345 7837-7859/courses.example.theanswer D/HostConnection: HostComposition ext ANDROID_EMU_CHECK
```

Install successfully finished in 104 ms.: App restart successful without requiring a re-install. (moments ago)

5. This is where all the debugging information from the app is displayed. It is a jumbled mess right now so let's filter it to only messages we are interested in.

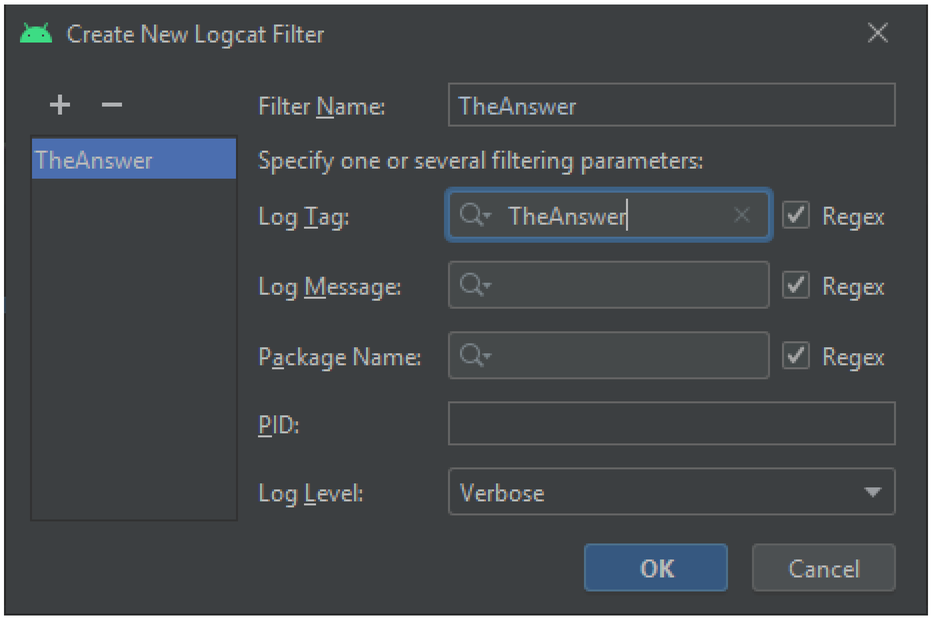
   On the right side of the Logcat pane, we can filter the messages to the ones that we want.


Logcat
Emulator Pixel_3_XL_API_30
course.examples.theanswer (11369)
Verbose
Regex
Edit Filter Configuration
Show only selected application
Firebase
No Filters
Edit Filter Configuration
logcat
Terminal
Build
6: Logcat
Profiler
4: Run
TODO
Event Log
Layout Inspector

```
2020-09-04 16:43:15.379 11369-11392/course.examples.theanswer W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
2020-09-04 16:43:15.386 11369-11392/course.examples.theanswer D/EGL_emulation: eglCreateContext: 0xf32694b0: maj 2 min 0 rcv 2
2020-09-04 16:43:15.403 11369-11392/course.examples.theanswer D/EGL_emulation: eglMakeCurrent: 0xf32694b0: ver 2 0 (tinfo 0xf35b3b10) (first time)
2020-09-04 16:43:15.424 11369-11392/course.examples.theanswer I/Gralloc4: mapper 4.x is not supported
2020-09-04 16:43:15.425 11369-11392/course.examples.theanswer D/HostConnection: createUnique: call
2020-09-04 16:43:15.425 11369-11392/course.examples.theanswer D/HostConnection: HostConnection::get() New Host Connection established 0xf3453e70, tid 11392
2020-09-04 16:43:15.425 11369-11392/course.examples.theanswer D/goldfish-address-space: allocate: Ask for block of size 0x100
2020-09-04 16:43:15.450 11369-11392/course.examples.theanswer D/goldfish-address-space: allocate: ioctl allocate returned offset 0x3f6ff6000 size 0x2000
2020-09-04 16:43:15.453 11369-11392/course.examples.theanswer D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync_v4 ANDROID_EMU_dma_v1 ANDROID_EMU_direct_mem ANDROID_
```

Install successfully finished in 160 ms.: App restart successful without requiring a re-install. (2 minutes ago)

6. Recall the 'TAG' const val we created in the 'companion object' was 'TheAnswer'. Create a Logcat filter with that tag.



Filter Name: TheAnswer

Specify one or several filtering parameters:

Log Tag: TheAnswer ☑ Regex

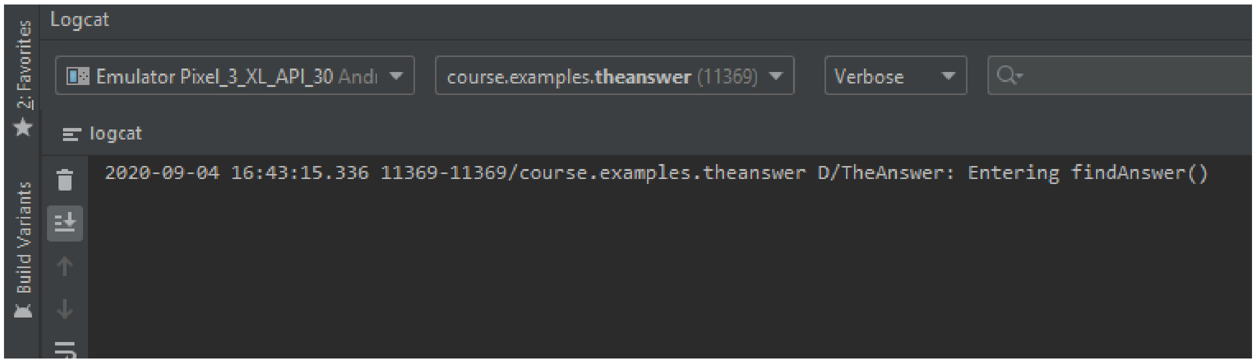
Log Message: ☑ Regex

Package Name: ☑ Regex

PID:

Log Level: Verbose

7. The Logcat should now filter out the other messages and only display the "Entering findAnswer()". Without changing the settings, modify the code to display a new line with your name when it runs.


2: Favorites
Build Variants
Logcat
Emulator Pixel_3_XL_API_30 Andı
course.examples.theanswer (11369)
Verbose
logcat

```
2020-09-04 16:43:15.336 11369-11369/course.examples.theanswer D/TheAnswer: Entering findAnswer()
```

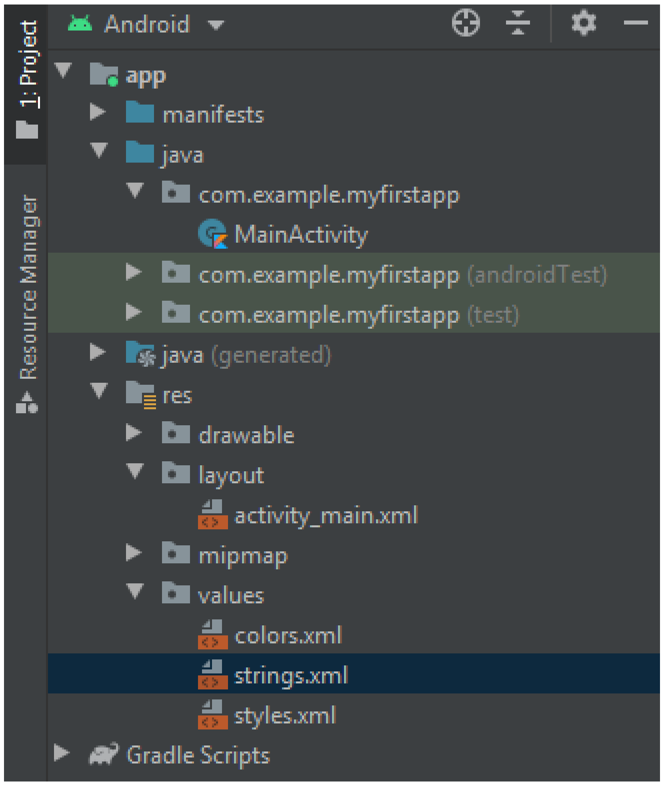
Extra Challenge

If you finish all the work above, try doing the challenge activity.

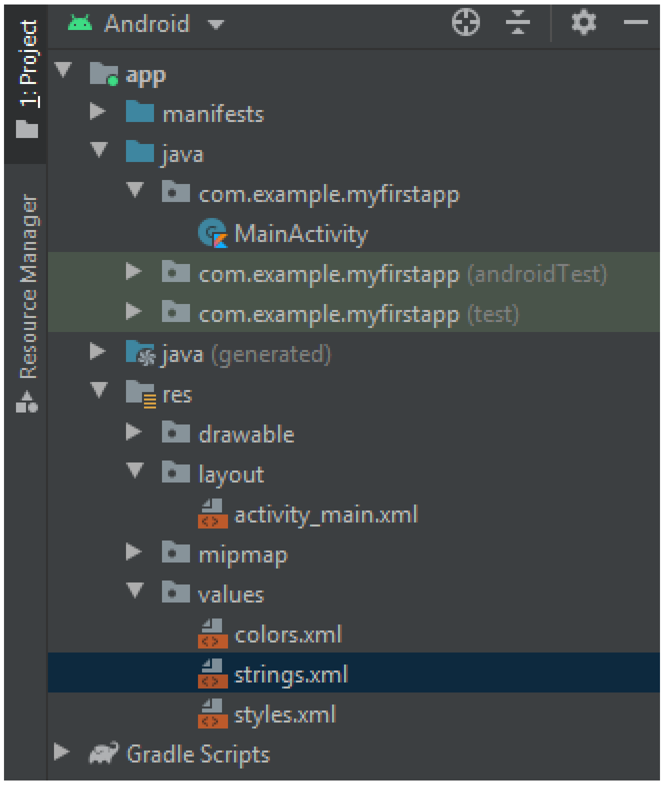
Modified Hello World

Remember the first app you made? Let's return to that! We will be modifying the original "Hello World!" message of your first app.

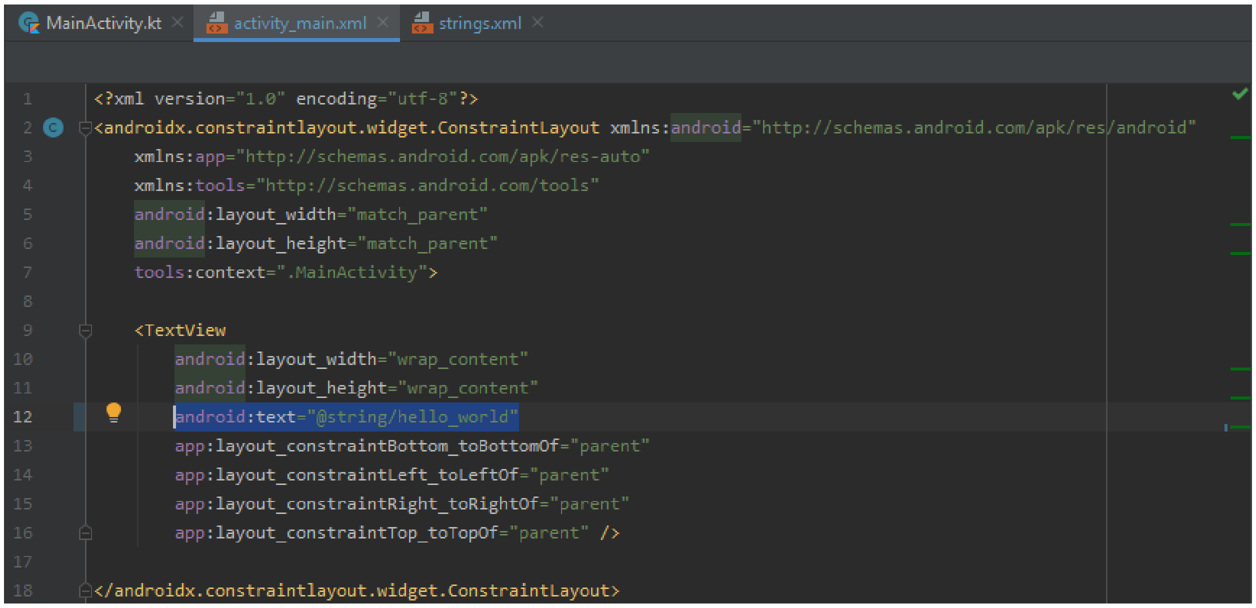
1. To do this, you need to modifiy the String value in '\res\values\strings.xml'.



2. Add another String element with the text: “My name is <your_name>!”.

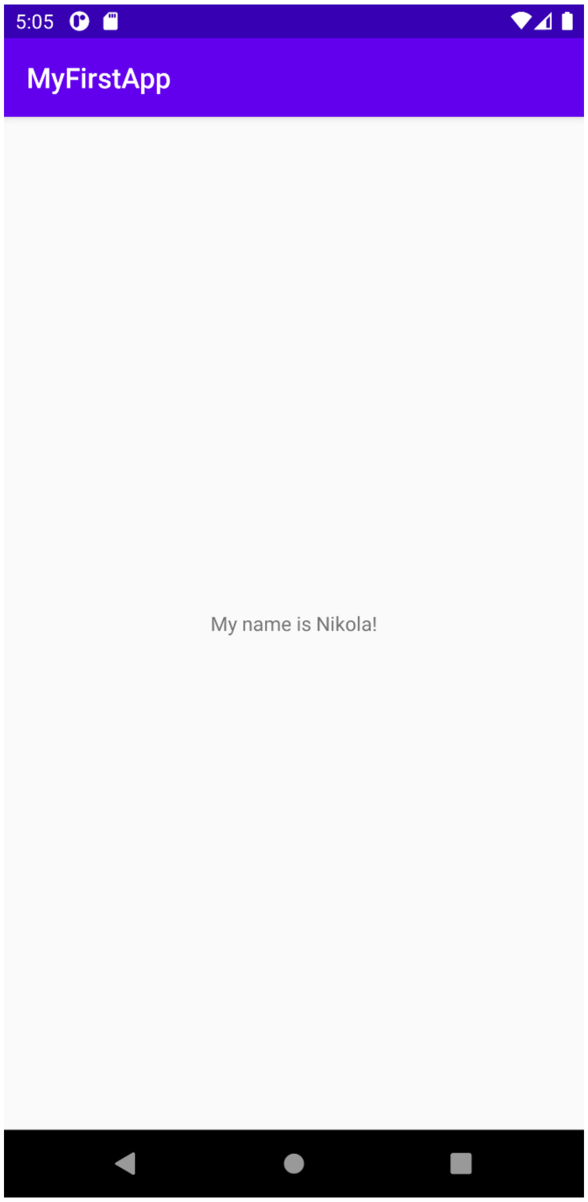


3. Now go to the ‘activity_main.xml’ file in ‘res/layout’. Edit the TextView element so that it references the hello_world String element you just created in the previous step.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

4. Now run the app and see the change.



For more information, take a look at:
https://developer.android.com/guide/topics/resources/string-resource.html

5. Now add support for another language such as Spanish! To do this, you will need to create an appropriate string file, run your app, change the emulator’s instance default language to Spanish, then rerun the app. Your Spanish String could be: “Hola Mundo! Me llamo <your_name>!”.

   For more information, take a look at:
   https://developer.android.com/guide/topics/resources/string-resource.html