CANONICAL

Developing Android apps on Ubuntu

August 2019

Executive summary

Android is the most popular mobile operating system and is continuing to grow its market share. IDC expects that Android will have 85.5% of the market by 2022[1], demonstrating that app development on Android will continue to be an in-demand skill.

Android Studio, the official Android development environment, provides an optimised intelligent code editor, a visual layout editor, hardware emulation for a variety of target devices, app and profiling and performance measurement tooling. It gives the software developer everything they need to build and test apps in an easy to use package.

Ubuntu is the perfect platform for Android app development with Android Studio. Using Ubuntu 18.04 LTS, developers have a secure and reliable desktop operating system which runs on a wide choice of laptops. Developers can be assured that their workstations are regularly updated with bug fixes, security patches and new features. Ubuntu users are free to upgrade their hardware such as adding additional memory, storage and GPU cards without any limits imposed by the operating system. In addition, there are no premium features locked behind upgrades or pay walls; all of Ubuntu's features are immediately available to all users.

Ubuntu features a wide variety of software development tools including numerous programming language compilers, integrated development environments (IDEs) and toolchains to enable developers to target Intel, ARM, Power, s390x, etc from their desktop. System developers building IoT solutions are able to quickly and easily work on software and debug using their preferred tools before building and deploying to power efficient hardware suited to IoT applications, all from their desktop.

Android Studio is available as a snap, the universal Linux app packaging format, making installation a breeze and provides full integration with physical Android hardware devices for testing as well as an emulator to allow testing on a multitude of target devices.

This guide will take developers through best-practice set up processes, writing an Android app and deploying it to emulated and physical devices using the standard tooling, all from their Ubuntu desktop computer.

[1]https://www.idc.com/promo/smartphone-market-share/os

Why develop on Ubuntu?

Android developers have access to the full suite of tools provided by Google in Android Studio including emulators and the intelligent code editor providing everything needed to develop an amazing new app, from the client to the server.

Every Long Term Support (LTS) release of Ubuntu comes with five years of free security and maintenance updates. Canonical also offers a number of additional products and services to help manage the security of your Ubuntu systems. All Canonical products are built with unrivalled security in mind – and tested to ensure they deliver it. Ubuntu software is secure from the moment you install it, and will remain so as Canonical ensures security updates are released quickly and frequently.

Ubuntu features a wide variety of software development tools including numerous programming language compilers, integrated development environments (IDEs) and toolchains to enable developers to target Intel, ARM, Power, s390x and other hardware from their desktop. System developers building IoT solutions are able to quickly and easily work on software and debug using their preferred tools before building and deploying to power efficient hardware suited to IoT applications, all from their desktop.

The same is true for cloud and server developers. Ubuntu is the most popular server instance in the cloud and by running Ubuntu Desktop developers can work, test and package software on their desktop before deploying to the cloud where the same underlying operating system is running. This gives developers a smooth path from development to production with the same version of software running on their desktop and target production environment.

Ubuntu users are free to upgrade their hardware to add additional memory, storage, GPU cards etc without any limits imposed by the operating system. There are no premium features locked behind upgrades or pay walls and all of Ubuntu's features are immediately available to all users.

Web developers have access to Node, Ruby, Python etc which are all native to the platform, and so again developers can test locally before deploying globally. Ubuntu has your favourite editors; Sublime, Atom, VS Code, gedit, etc. Ubuntu ships the latest browsers from Mozilla Firefox and Google Chrome, as well as Opera, Chromium & Brave. Ubuntu users can even test on Internet Explorer and Edge using official virtual machines freely available from Microsoft.

Creating Docker images is easy as well, since the underlying platform is the same. You can develop and debug on your desktop or laptop before moving to a container for deployment. The packages and libraries are the same in both environments.

Team communication tools such as Slack and Skype are available to install as snaps direct from the vendors, meaning it's easy to stay in touch.

Getting started

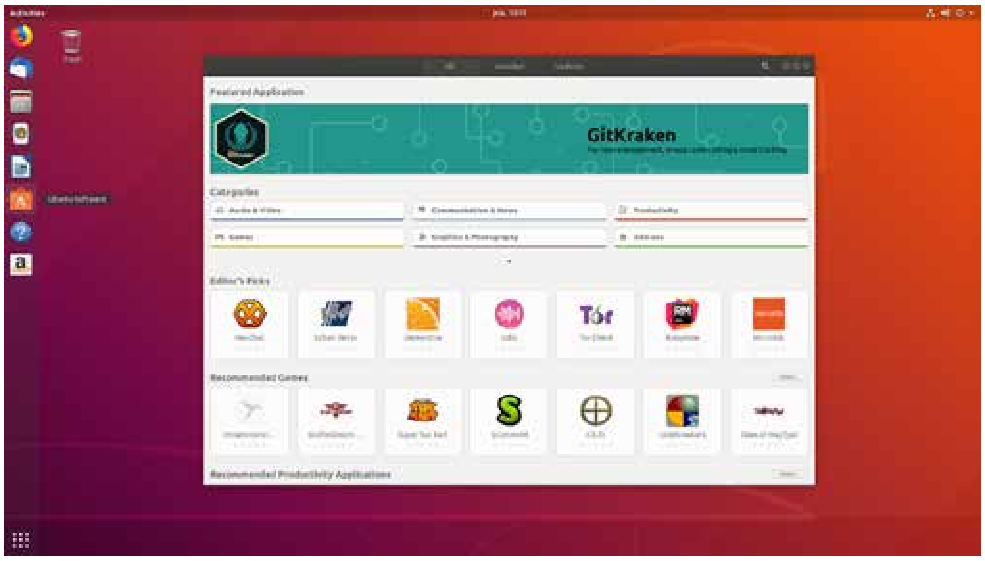
Here we layout the steps needed to install Android Studio to start writing your first app. Android Studio can be installed as a snap, meaning that it will be automatically kept up to date when a new version is released. Snaps also have the benefit of containing applications within a sandbox, and although Android Studio is packaged as a 'classic' snap – meaning that it has full access to your file system and hardware, needed for hardware integration etc – the installed files are kept together in the snap, not unpacked all over your disk.

Installing Android Studio

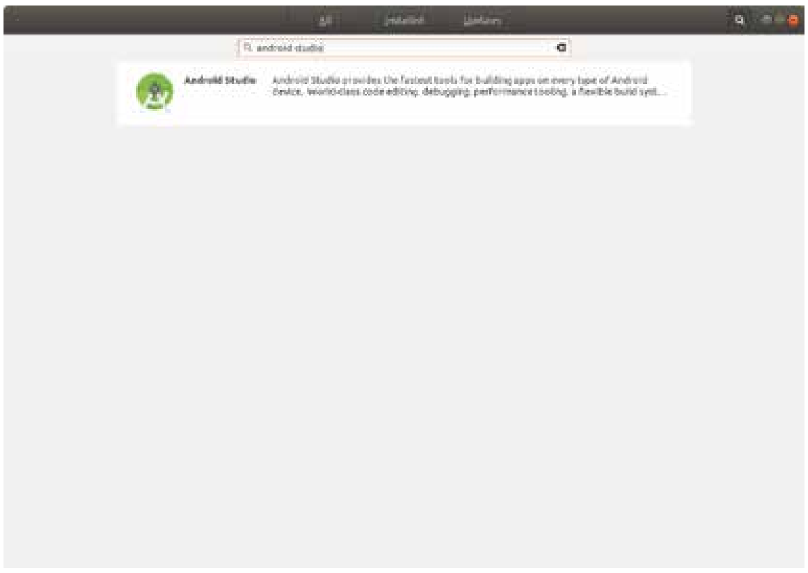
Installing Android Studio is very straight forward. You can locate it in the Ubuntu Software store, along with thousands of other apps, games and utilities. Ubuntu Software also handles firmware updates for supported platforms, keeping your computer up to date. Finding the software you are looking for is as easy as searching. Open Ubuntu Software and just type 'android studio'. From the command line type:

```
$ snap install android-studio
```

You can also install via the online snap store by opening this link in your browser: https://snapcraft.io/android-studio which will then take you in to Ubuntu Software as below.



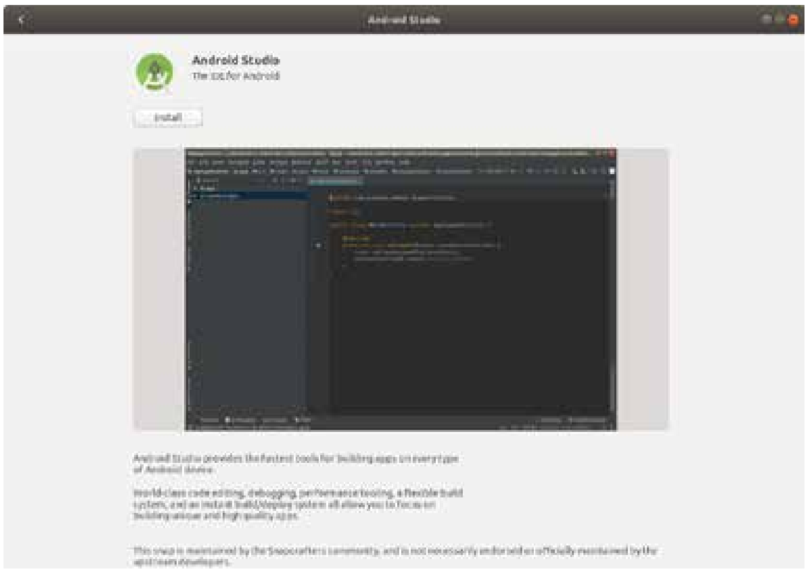
Ubuntu Software will quickly search through thousands of apps to find you the most relevant results:



Android Studio

Android Studio provides the fastest tools for building apps on every type of Android device. World-class code editing, debugging, performance tooling, a flexible build syst...

Click on the Android Studio entry, and click 'Install':



Android Studio

The IDE for Android

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

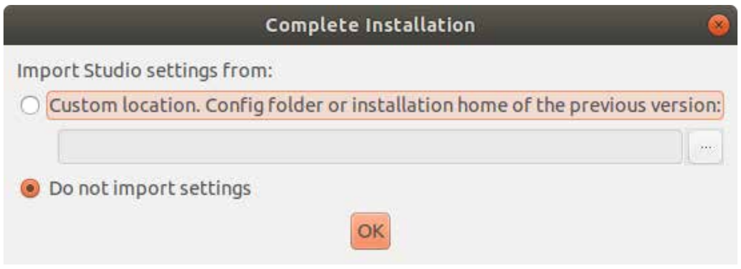
World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps.

This snap is maintained by the Snapcrafters community, and is not necessarily endorsed or officially maintained by the upstream developers.

That's all there is to it. Once Android Studio is installed you are ready to go.

Configuration of Android Studio

The first time you start Android Studio you will be prompted to import any settings from previous installations. We're assuming that you haven't installed Android Studio before, so this can be ignored. Choose 'Do not import settings' and click 'OK':

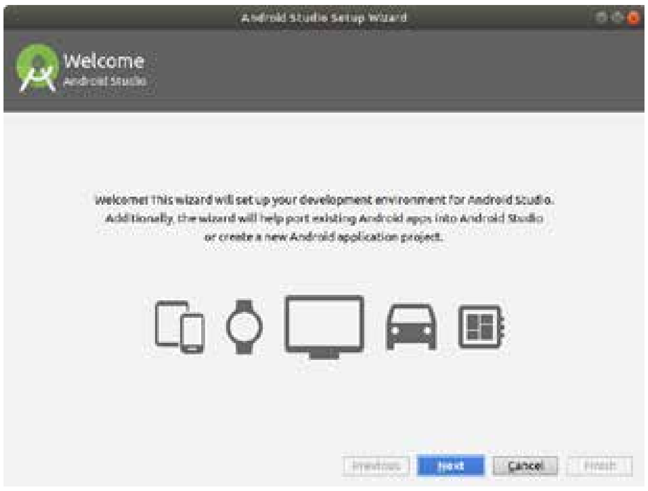


Import Studio settings from:

- [ ] Custom location. Config folder or installation home of the previous version:
- [✓ Do not import settings]

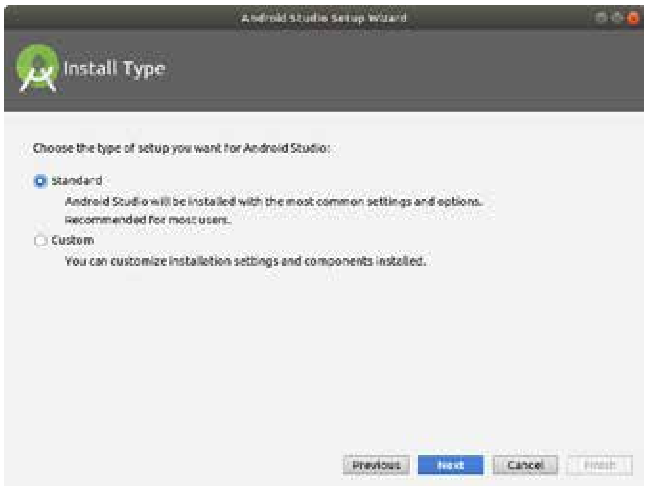
Setting up

When you run Android Studio for the first time, the Setup Wizard will help guide you through configuration. Documented below are the steps to follow to get Android Studio working and how to set up the emulation needed for testing on virtual devices.



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.

Selecting the 'Standard' type of setup is a good place to start. If you have more experience with Android Studio you can explore the 'Custom' options, but 'Standard' is a good set of defaults.

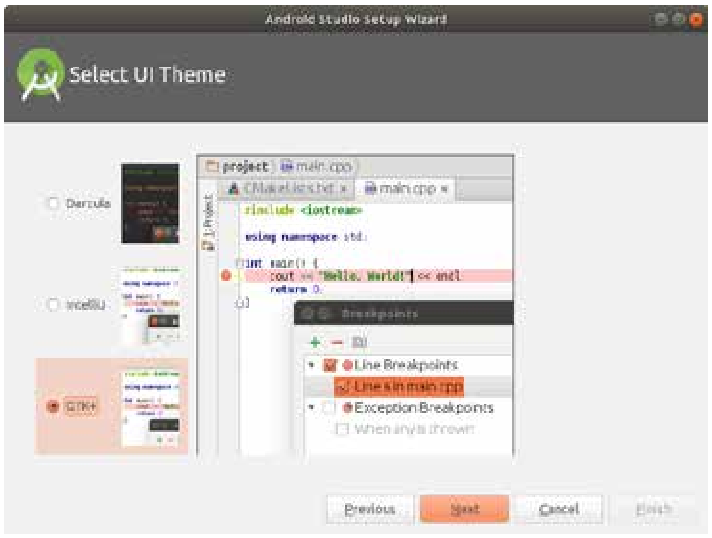


Choose the type of setup you want for Android Studio:

[✓ 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.

In order to get Android Studio looking great on Ubuntu you should select the GTK+ UI theme. This means that your desktop theme, colour scheme, font selection and other elements will be reflected in Android Studio as well.



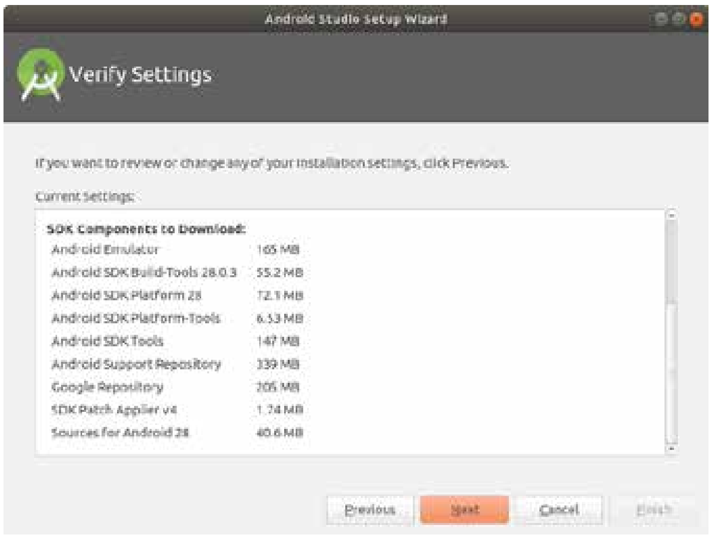
```
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, World!" << endl
    return 0;
}
```

The next screen shows what Android Studio is going to download and prepare. The default setup is approximately 1GB and contains notably:

- The Android Emulator machinery
- Latest SDK (build tools, runtime, debugger, documentation)
- Some common libraries like the Android Support Repository (to support previous versions of Android in your application while targeting latest SDK) and Google services
- Android platform source code



If you want to review or change any of your installation settings, click Previous.

Current Settings:

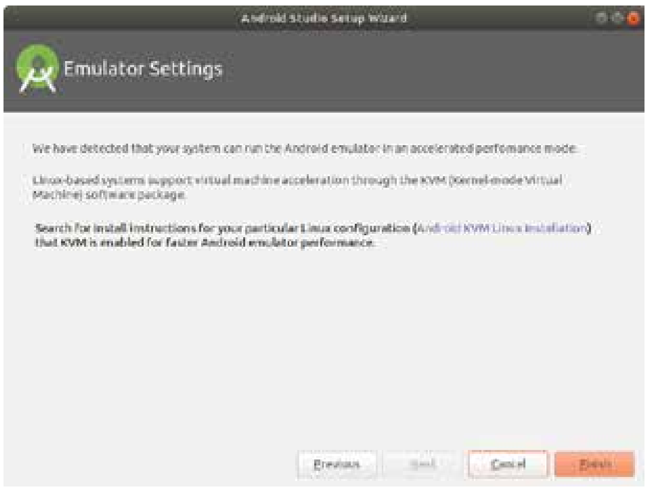
SDK Components to Download:

Android Emulator	165 MB
Android SDK Build-Tools 28.0.3	55.2 MB
Android SDK Platform 28	72.1 MB
Android SDK Platform-Tools	6.53 MB
Android SDK Tools	147 MB
Android Support Repository	339 MB
Google Repository	205 MB
SDK Patch Applier v4	1.74 MB
Sources for Android 28	40.6 MB

Hardware acceleration for your emulator

If your system supports virtualisation via CPU virtualisation features, the Android emulator can run in accelerated performance mode. This means graphics, network, CPU and memory specific calls can directly go down to your hardware bringing a performance boost.

This screen will give more information if this is the case, and ask you to set up the virtualisation technology:



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

Linux-based systems support virtual machine acceleration through the KVM (Kernel-mode Virtual Machine) software package.

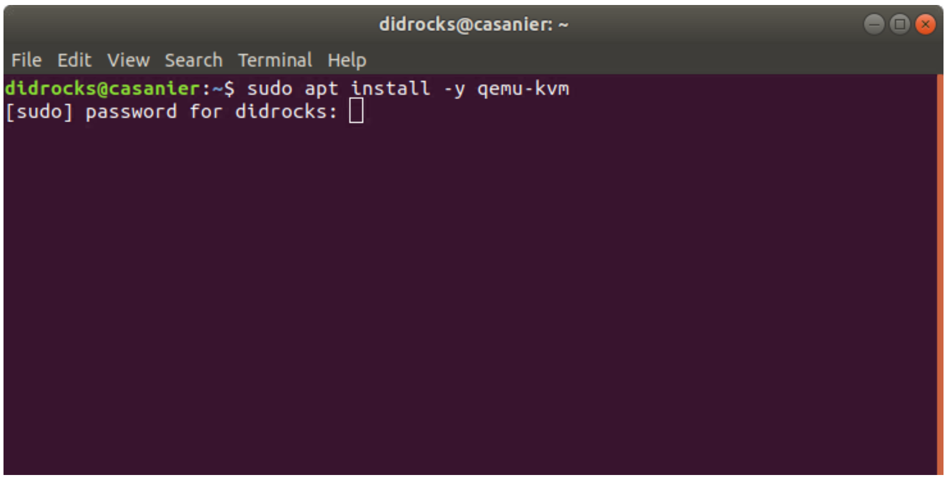
Search for install instructions for your particular Linux configuration (Android KVM Linux Installation) that KVM is enabled for faster Android emulator performance.

To get this set up for Ubuntu, follow these steps:

- Open a terminal (press the super key and search for 'terminal')
- Type:

```
sudo apt install -y qemu-kvm
```

- Your password will be requested, type it here and press enter (no characters will be printed while you type it).



```
didrocks@casanier:~$ sudo apt install -y qemu-kvm
[sudo] password for didrocks: 
```

Once downloaded and installed you can type:

```
sudo kvm-ok
```

If everything is set up correctly, you will see:

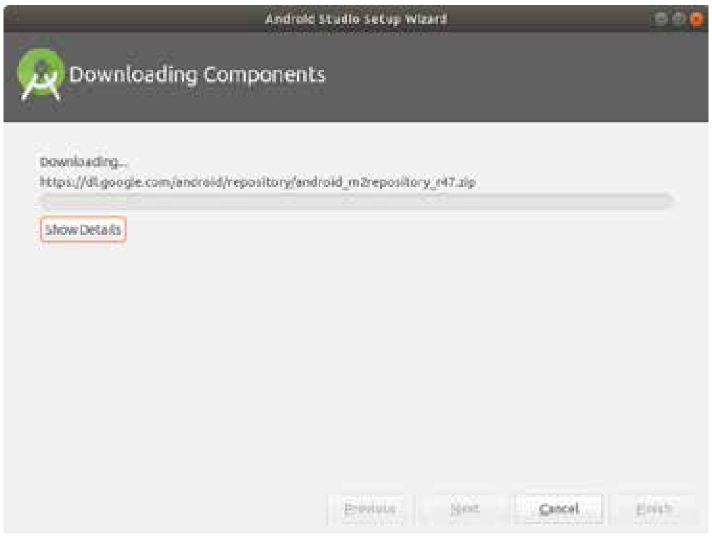
```
INFO: /dev/kvm exists
KVM acceleration can be used
```

If you do not see that KVM acceleration can be used, there are likely two options:

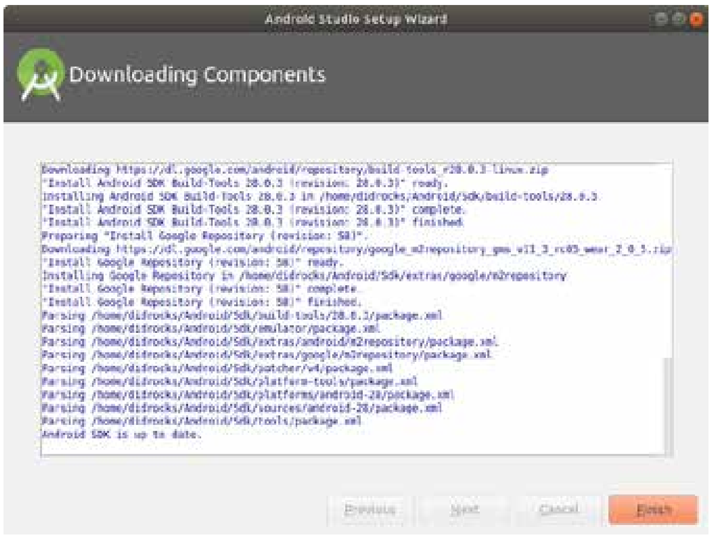
- You need to enable hardware acceleration in your BIOS or UEFI system on boot. This is a hardware dependant option, please refer to your corresponding system documentation.
- Your system doesn't support hardware acceleration. This is the case, for example, on older hardware or if you are using Ubuntu in a Virtual Machine (nesting VM hardware acceleration doesn't work). Remember that the Android emulator is itself a virtual machine.

More information on enabling hardware acceleration for Android Studio emulator is available here.

Click on 'Finish' and wait for the selected components to download and install. This may take a while depending on your network connection.



```
Downloading...
https://dl.google.com/android/repository/android_m2repository_r47.zip
```



```
Downloading https://dl.google.com/android/repository/build-tools_r28.0.3-linux.zip
"Install Android SDK Build-Tools 28.0.3 (revision: 28.0.3)" ready.
Installing Android SDK Build-Tools 28.0.3 in /home/didrocks/Android/Sdk/build-tools/28.0.3
"Install Android SDK Build-Tools 28.0.3 (revision: 28.0.3)" complete.
"Install Android SDK Build-Tools 28.0.3 (revision: 28.0.3)" finished.
Preparing "Install Google Repository (revision: 58)".
Downloading https://dl.google.com/android/repository/google_m2repository_gms_v11_3_rc05_wear_2_0_5.zip
"Install Google Repository (revision: 58)" ready.
Installing Google Repository in /home/didrocks/Android/Sdk/extras/google/m2repository
"Install Google Repository (revision: 58)" complete.
"Install Google Repository (revision: 58)" finished.
Parsing /home/didrocks/Android/Sdk/build-tools/28.0.3/package.xml
Parsing /home/didrocks/Android/Sdk/emulator/package.xml
Parsing /home/didrocks/Android/Sdk/extras/android/m2repository/package.xml
Parsing /home/didrocks/Android/Sdk/extras/google/m2repository/package.xml
Parsing /home/didrocks/Android/Sdk/patcher/v4/package.xml
Parsing /home/didrocks/Android/Sdk/platform-tools/package.xml
Parsing /home/didrocks/Android/Sdk/platforms/android-28/package.xml
Parsing /home/didrocks/Android/Sdk/sources/android-28/package.xml
Parsing /home/didrocks/Android/Sdk/tools/package.xml
Android SDK is up to date.
```

Once that process has finished click and 'Finish'.

Android Studio is now installed along with accelerated emulation for testing. Remember that you will always be running the latest Android Studio release, updated automatically via the snap.

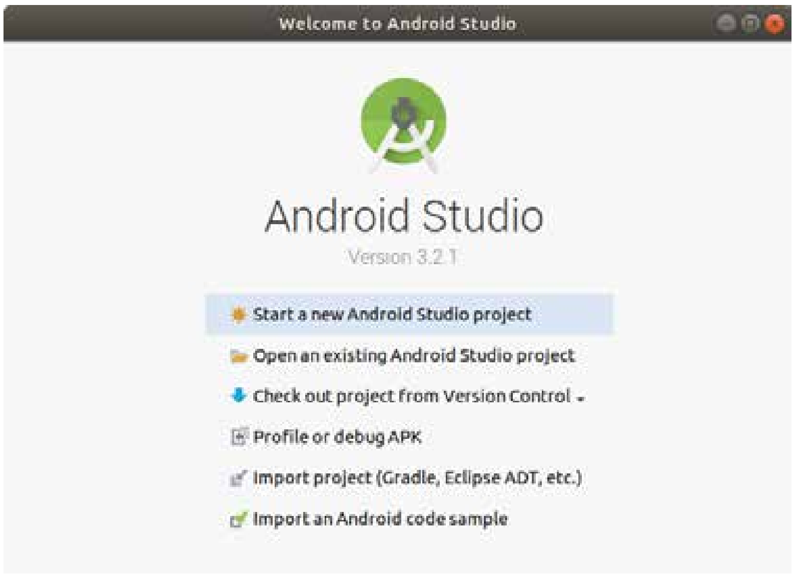
Scaffolding your first application and running it in the emulator.

Time for your first application. We'll use the sample Hello World provided by Android Studio.

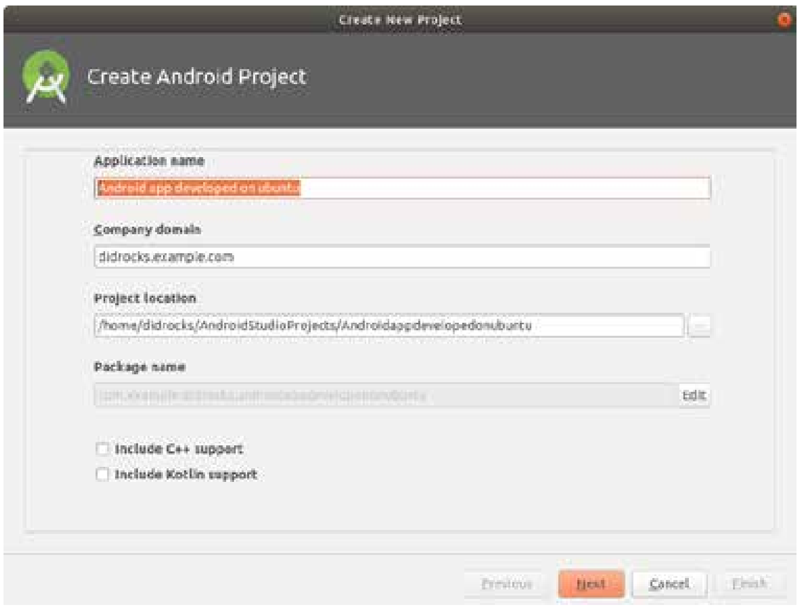
Creating your application

The Android Studio Welcome Wizard will help import or create any new Android application project.

Start a new Android Studio project by selecting the corresponding option.


Welcome to Android Studio
Android Studio
Version 3.2.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample

Choose an application name, like 'Android app developed on Ubuntu', a company domain (you can keep the generated default for this experiment), and keep the default project location. Create a simple Java application and don't select C++ or Kotlin support for now. Click 'Next'.



Application name
Android app developed on ubuntu

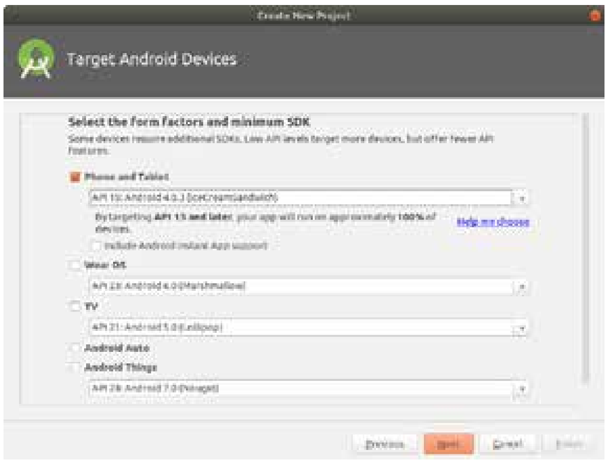
Company domain
didrocks.example.com

Project location
/home/didrocks/AndroidStudioProjects/Androidappdevelopedonubuntu

Package name
[illegible]

- [ ] Include C++ support
- [ ] Include Kotlin support

We are going to create a phone application, based on API 15, to be compatible with Android 4.0 and later, which covers the vast majority of devices. When you select different API target levels, you can see the percentage of devices in the market you will reach. Click ‘Next’.


Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

[✓] **Phone and Tablet**
API 15: Android 4.0.3 (IceCreamSandwich)
By targeting **API 15 and later**, your app will run on approximately **100%** of devices.
Help me choose
[ ] Include Android Instant App support

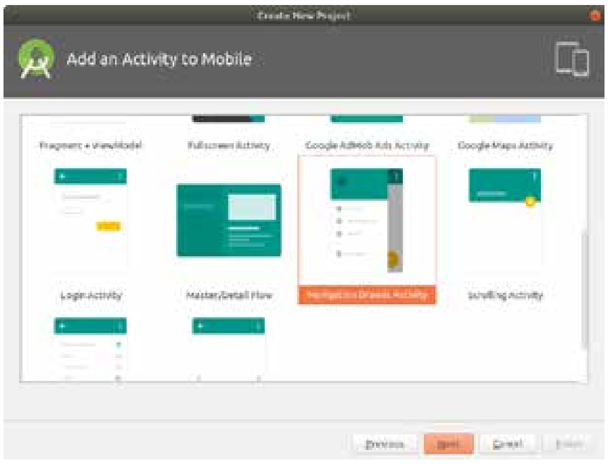
[ ] **Wear OS**
API 23: Android 6.0 (Marshmallow)

[ ] **TV**
API 21: Android 5.0 (Lollipop)

[ ] **Android Auto**

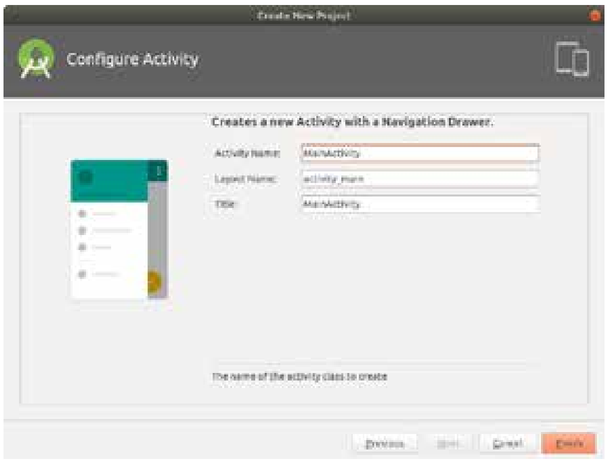
[ ] **Android Things**
API 24: Android 7.0 (Nougat)

As an example, select ‘Navigation Drawer Activity’ as the first application activity. Click ‘Next’.


Create New Project

Add an Activity to Mobile

Keep the default activity names:


Create New Project

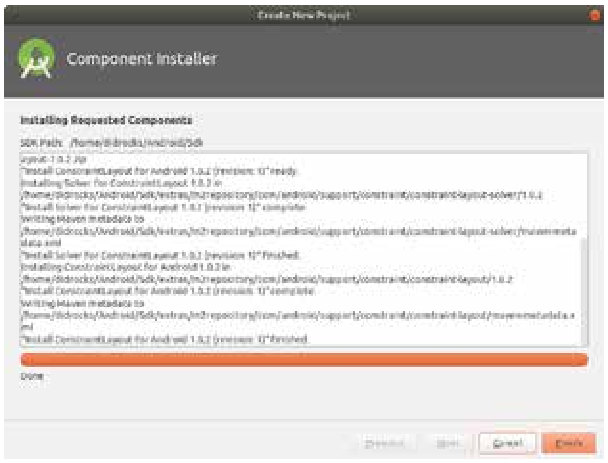
Configure Activity

Creates a new Activity with a Navigation Drawer.

Activity Name:	MainActivity
Layout Name:	activity_main
Title:	MainActivity

The name of the activity class to create

Click ‘Finish’ and any additional components required by your application will be installed:


Component Installer

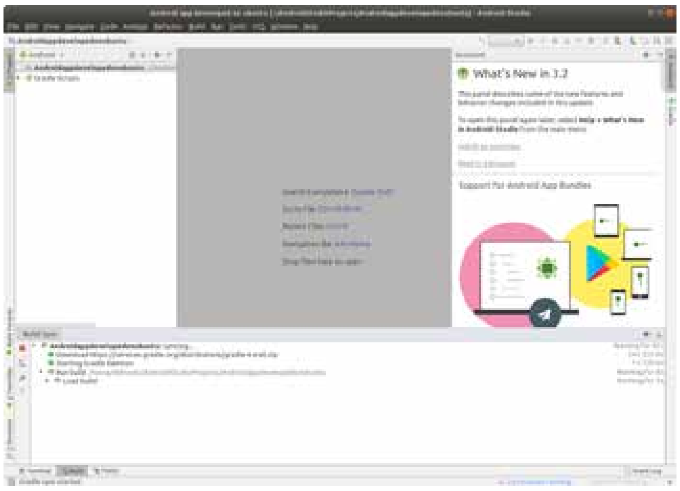
Installing Requested Components

SDK Path: /home/didrocks/Android/Sdk

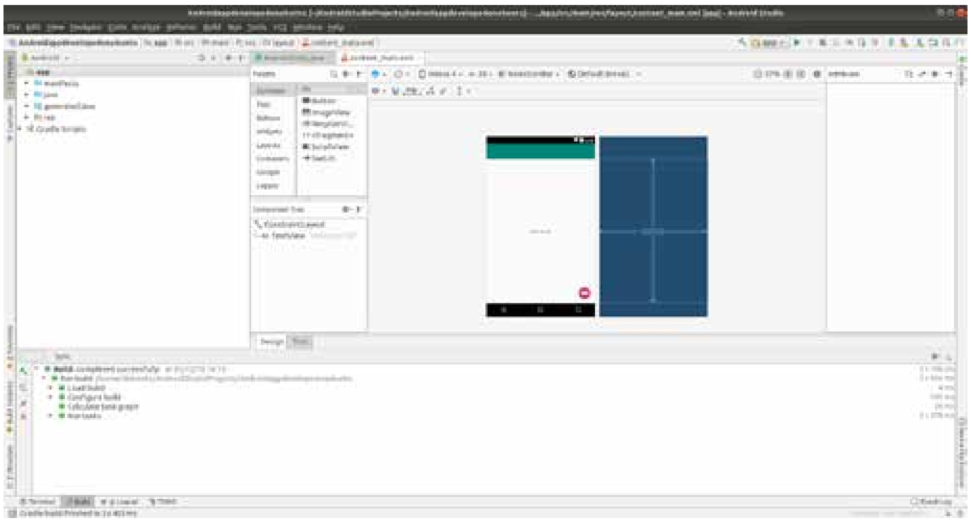
```
ayout-1.0.2.zip
"Install ConstraintLayout for Android 1.0.2 (revision: 1)" ready.
Installing Solver for ConstraintLayout 1.0.2 in
/home/didrocks/Android/Sdk/extras/m2repository/com/android/support/constraint/constraint-layout-solver/1.0.2
"Install Solver for ConstraintLayout 1.0.2 (revision: 1)" complete.
Writing Maven metadata to
/home/didrocks/Android/Sdk/extras/m2repository/com/android/support/constraint/constraint-layout-solver/maven-meta
data.xml
"Install Solver for ConstraintLayout 1.0.2 (revision: 1)" finished.
Installing ConstraintLayout for Android 1.0.2 in
/home/didrocks/Android/Sdk/extras/m2repository/com/android/support/constraint/constraint-layout/1.0.2
"Install ConstraintLayout for Android 1.0.2 (revision: 1)" complete.
Writing Maven metadata to
/home/didrocks/Android/Sdk/extras/m2repository/com/android/support/constraint/constraint-layout/maven-metadata.x
ml
"Install ConstraintLayout for Android 1.0.2 (revision: 1)" finished.
```

Done

In the background, Gradle, the Android application build system, will be downloaded, configured and a first build of your project will run.

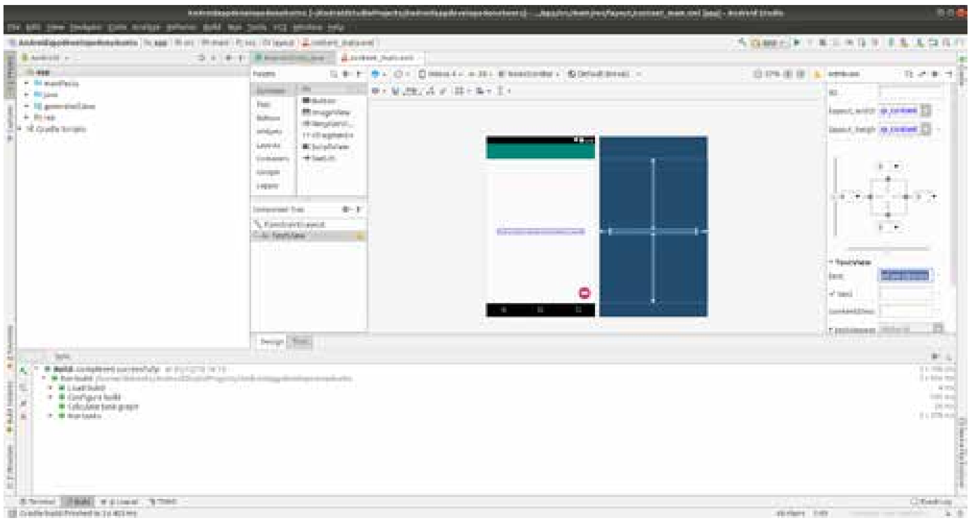


Once the build is done, you will see the .xml editor of your application showing the UI.



Application modification

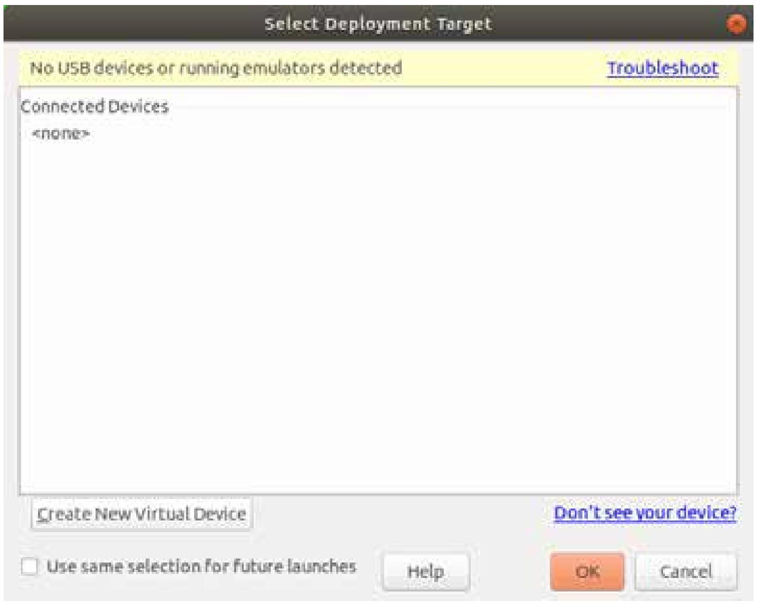
As an example, modify the 'Hello World!' text and change it to 'My first Android Application developed on Ubuntu'. Double click on the text and change the content in the TextView, then press 'Enter'.



Setup the emulator

Time to setup the application in the emulator. Run the application by clicking on the 'Run' icon in the upper right toolbar or by simply pressing Shift + F10.

You will be prompted to select your device to target, but right now there aren't any.



Select Deployment Target

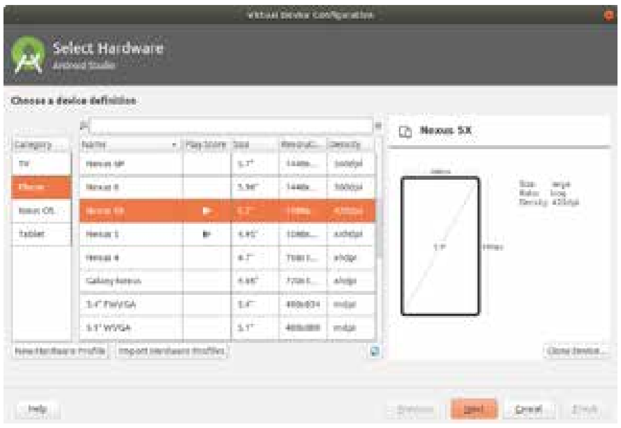
No USB devices or running emulators detected   Troubleshoot

Connected Devices

<none>

Create New Virtual Device
Don't see your device?
Use same selection for future launches
Help
OK
Cancel

Create an emulator by clicking on the 'Create New Virtual Device' button. You will be presented with a list of pre-configured devices. This makes it easy to select the size, screen resolution and pixel density for common phones, TVs and tablet hardware. Select a 'Nexus 5X'.


Virtual Device Configuration

Select Hardware
Android Studio

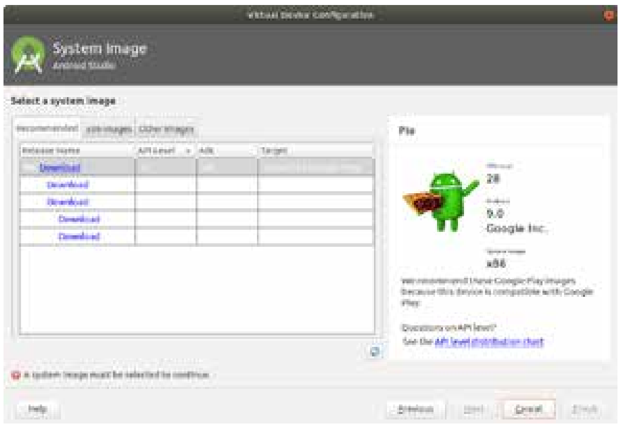
Choose a device definition

Category
TV
Phone
Wear OS
Tablet

Name	Play Store	Size	Resolution	Density
Nexus 6P		5.7"	1440x...	560dpi
Nexus 6		5.96"	1440x...	560dpi
Nexus 5X	▶	5.2"	1080x...	420dpi
Nexus 5	▶	4.95"	1080x...	xxhdpi
Nexus 4		4.7"	768x1...	xhdpi
Galaxy Nexus		4.65"	720x1...	xhdpi
5.4" FWVGA		5.4"	480x854	mdpi
5.1" WVGA		5.1"	480x800	mdpi

Nexus 5X

The next screen allows you to specify which version of Android will be installed on to the virtual phone. Some of these have Google Play services installed and some don't. As a general rule, select the newest version of Android you can for the target devices. Click on 'Download' on the first line to get the latest version with Google Play.


Virtual Device Configuration

System Image
Android Studio

Select a system image

Release Name	API Level	ABI	Target
Download			
Download			
Download			
Download			
Download			

Pie

API Level
28

Android
9.0
Google Inc.

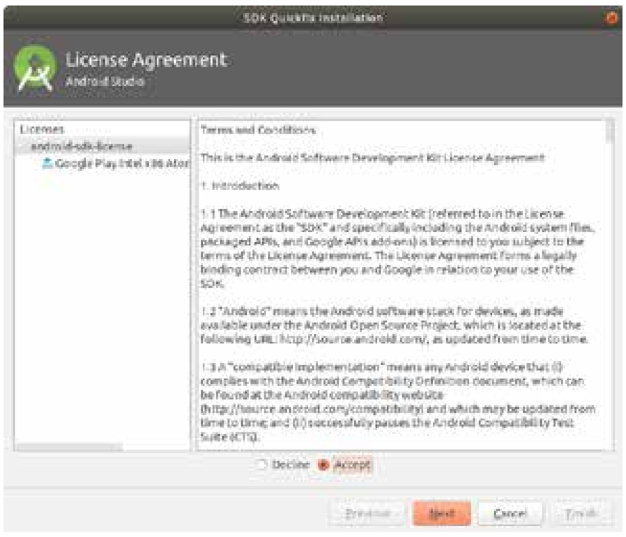
System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the API level distribution chart

A system image must be selected to continue.

You need to accept the license to download it.


SDK Quickfix Installation

License Agreement
Android Studio

Licenses
android-sdk-license
Google Play Intel x86 Atom

Terms and Conditions

This is the Android Software Development Kit License Agreement

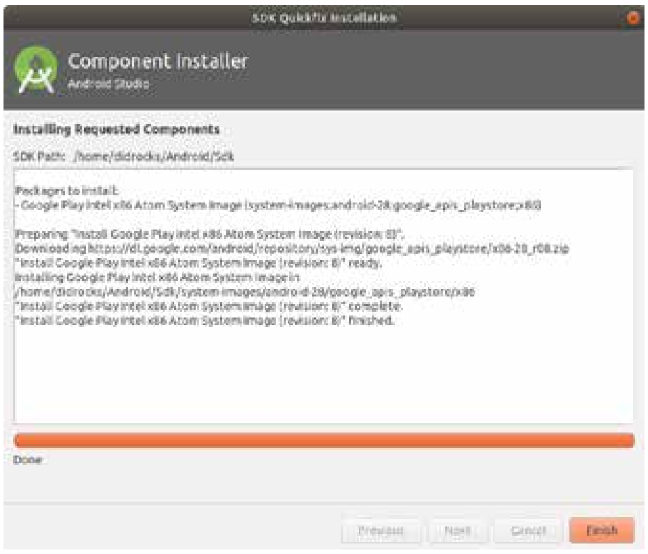
1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.

1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).

Decline [✓ Accept]



Installing Requested Components

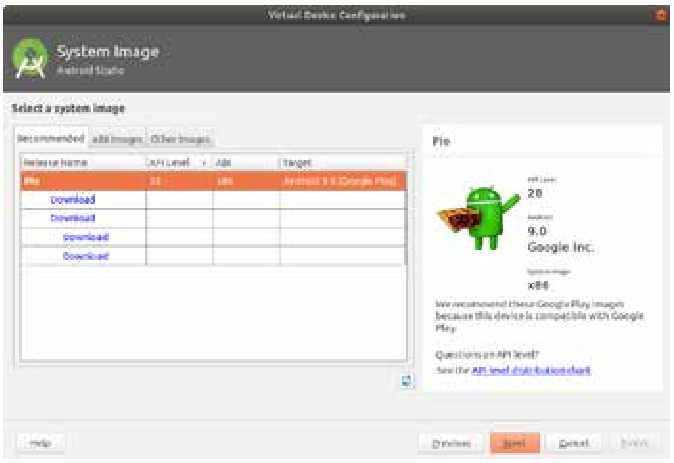
SDK Path: /home/didrocks/Android/Sdk

```
Packages to install:
- Google Play Intel x86 Atom System Image (system-images;android-28;google_apis_playstore;x86)

Preparing "Install Google Play Intel x86 Atom System Image (revision: 8)".
Downloading https://dl.google.com/android/repository/sys-img/google_apis_playstore/x86-28_r08.zip
"Install Google Play Intel x86 Atom System Image (revision: 8)" ready.
Installing Google Play Intel x86 Atom System Image in
/home/didrocks/Android/Sdk/system-images/android-28/google_apis_playstore/x86
"Install Google Play Intel x86 Atom System Image (revision: 8)" complete.
"Install Google Play Intel x86 Atom System Image (revision: 8)" finished.
```

Done

Keeping the system image you just downloaded selected, in our case, Android Pie, API Level 28 (corresponding to Android 9.0) and click 'Next'.



Select a system image

Release Name	API Level	ABI	Target
Pie	28	x86	Android 9.0 (Google Play)
Download			
Download			
Download			
Download			

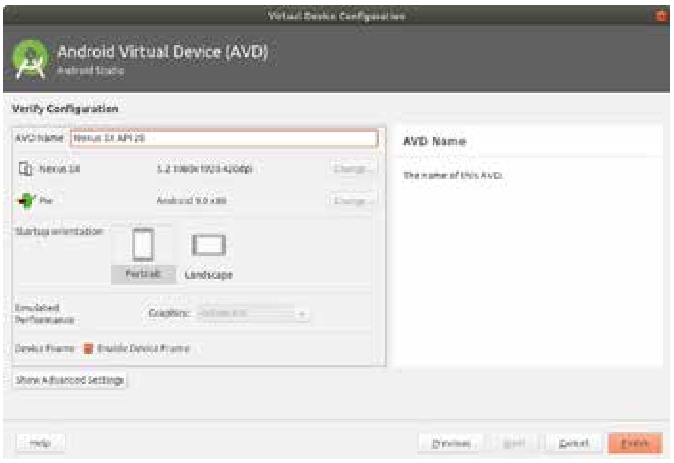
Pie

API Level: 28
Android: 9.0
Google Inc.
System Image: x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the API level distribution chart

Check the emulator configuration (device and android version, default orientation) and click 'Finish' to add your new virtual devices emulated system.



Verify Configuration

AVD Name: Nexus 5X API 28

Nexus 5X — 5.2 1080x1920 420dpi

Pie — Android 9.0 x86

Startup orientation: Portrait

Emulated Performance — Graphics: Automatic

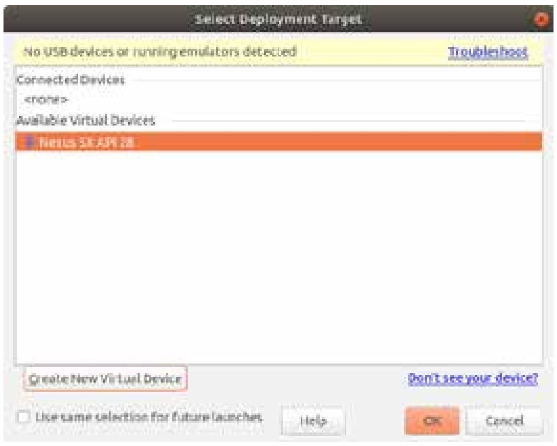
Device Frame: Enable Device Frame

AVD Name

The name of this AVD.

Try your application in the emulator

We you have a virtual Nexus 5X, API 28 available to run your app. Ensure it's selected and click 'OK'. (NB: later in this process you will also test on a real physical phone, so do not check 'Use same selection for future launches')



Select Deployment Target

No USB devices or running emulators detected Troubleshoot

Connected Devices

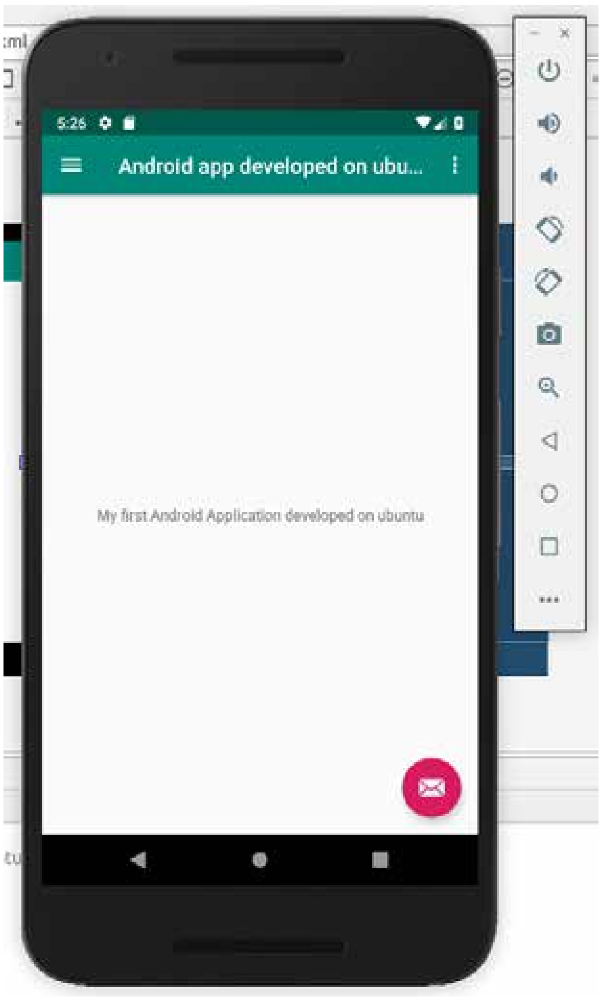
<none>

Available Virtual Devices

Nexus 5X API 28

Create New Virtual Device
Don't see your device?
Use same selection for future launches
Help
OK
Cancel

The emulator starts, Android boots, and your application is started.



My first Android Application developed on ubuntu

You have a control panel with physical buttons emulations on the right side. Take some time to play with your application and toggle the application drawer.

Setup your physical Android device

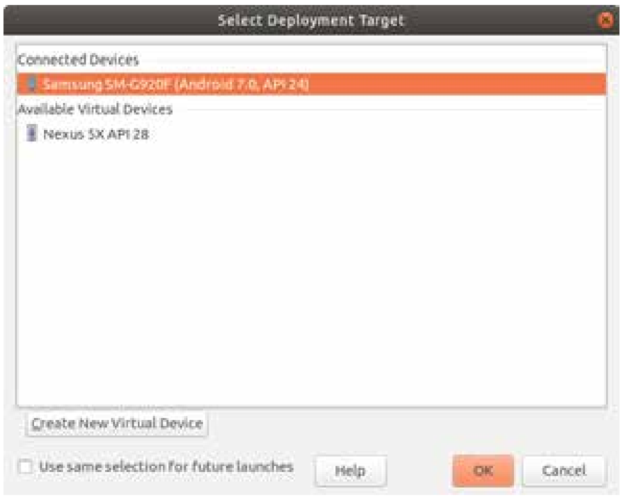
The next stage is to try your app on a real Android device.

The first step is to enable developer mode on your chosen device. Depending on your device brand and Android version, this process can vary. On most devices you go to Settings -> About phone -> and click 7 (seven) times on the serial number. You should see notifications for the last 3 taps.

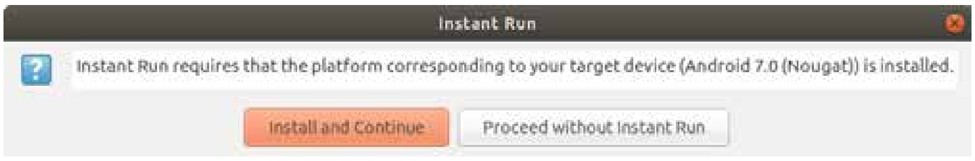
If successful, then a new 'Developer Options' entry in available in the 'Settings' window. Ensure it's enabled and that USB debug is switched on as well. Then plug your physical device to your laptop via a USB cable. A permission request is displayed on your phone the first time you connect it. You will need to accept it.

Try your application on your Android device

To test on real hardware click on the 'Run' button in Android Studio (or Shift + F10) on your desktop. This time, your Connected Devices should appear in addition to the Virtual Device we previously created. Select it and click 'OK'.


Select Deployment Target
Connected Devices
Samsung SM-G920F (Android 7.0, API 24)
Available Virtual Devices
Nexus 5X API 28
Create New Virtual Device
Use same selection for future launches
Help
OK
Cancel

You can optionally install 'Instant Run' on your target device. This feature enables you to re-run the application more quickly by not sending the entire app to the phone for every change. For now, do not use this option. You can enable it later if everything works as expected.



Instant Run requires that the platform corresponding to your target device (Android 7.0 (Nougat)) is installed.

Your application should run on your phone and appear on the screen. You can test it to see that screen rotates and other phone features integrate properly. If you navigate to your phone application launcher UI, you will see 'Android Application developed on Ubuntu' installed as any with any other application. You can disconnect the phone from your computer to continue testing.

Once you are done, you can quit it like any other application or you can use the 'Stop' icon from Android Studio if the phone is still connected. Note that the application will stay installed on your phone and you can uninstall it as with any other app.

Conclusion

This is only the start of the journey, there is a lot more features that Android Studio provides such as remote debugging, layout management, older version support for newer features and more. The Android Studio user guide is the definitive place to learn more about Android development, and you are now ready to dive deeper, developing Android Applications on your Ubuntu system, testing locally in a virtual environment simulating many different devices and form factors or using a real device.

Ubuntu desktop provides you a solid, reliable base on which to build your Android apps using all the official tools from Google. The tooling will be kept up-to-date automatically, and the OS will benefit from five years of security and bug fix updates.

Further resources:

Android developer blog
Android developer backstage (official Google podcast)
Install Ubuntu desktop

ubuntu®
Delivered by Canonical