Gradle 与Android Studio（AS）全面解析

# Gradle

# Bintray

Bintray是jCenter的提供商，他支持上传lib到多个平台，jCenter只是众多平台中的一个，形象的说jCenter是位于某地的仓库，Bintray是送货的卡车，你写的库就是货了。如果我们要想把自己写的lib分享给其他人，就必须通过Bintray上传到jCenter。首先我们要去[Bintray](https://link.jianshu.com/?t=https://bintray.com/" \t "http://www.jianshu.com/p/_blank)上注册一个账号，或者用Github帐号直接登录。

# 模块和应用的gradle关系

# Gradle.wrapper

Gradle 不断的在发展，新的版本难免会对以往的项目有一些向后兼容性的问题，这个时候,gradle wrapper就应运而生了。gradlw wrapper 包含一些脚本文件和针对 不同系统下面的运行文件。wrapper 有版本区分，但是并不需要你手动去下载，当你运行脚本的时候，如果本地没有会自动下载对应版本文件。

* 1. Gradle-wrapper.jar
  2. Gradle-wrapper.properties

# Build.gradle配置

# compileSdkVersion

是告诉gradle用哪个SDK版本来编译。和运行时要求的版本号没有关系。compileSDK只是关系到你编译出来的包。不是运行的表现行为。所以修改compileSdkVersion是不会改变你app在手机上运行的表现行为的。但是我们一般情况，也是最理想的情况就是把compileSdkVersion设置到最高，因为使用新的编译检查可以获得很多好处，可以避免弃用的API，并且为使用新的API做好准备。如果出现如下错误：

resource found that matches the given name

'android:Widget.Material.Spinner.Underlined'.

将compileSdkVersion修改为最高版本，可以解决此问题。

# minSdkVersion

序运行的最低的要求的Sdk，如果系统的API level低于android:minSdkVersion设定的值，那么android系统会阻止用户安装这个应用。例如你设置miniSdk是3.0的话，你写的方法只有在4.0后才有的方法，这个时候就会提示你在3.0的时候是没有这个方法的。



解决办法是判断SDK版本，例如

Build.VERSION.SDK\_INT >=Build.VERSION\_CODES.M。

# maxSdkVersion

如果应用设置的maxSdkVersion 值低于系统本身使用的API Level，系统将不会允许安装该应用。在系统升级后，新系统会重新校验这个值，如果新系统的API Level高于这个值，新系统会删除你的应用。

# targetSdkVersion

为了让你的应用程序支持每个Android版本，你应当提高targetSdkVersion的值到最新的API level，然后在对应的平台上彻底的测试你的应用。targetSdkVersion这个属性是在程序运行时期起作用的，系统根据这个属性决定要不要以兼容模式运行这个程序。

minSDK和targetSDK，这两者相当于一个区间。你能够用到targetSDK中最新的API和最酷的新功能，但你又不得不向下兼容到minSDK，保证这个区间内的设备都能够正常的执行你的app。换句话说，你想使用Android刚刚推出的新特性。但这对于你的app又不是必须的。你就能够将targetSDK设置为你想使用新特性的SDK版本号，minSDK设置成低版本号保证全部人都能够使用你的app。

# buildToolsVersion

在SDK Manager中安装。buildToolsVersion版本不能小于compileSdkVersion.

* + 1. defaultConfig
    2. Krepositories
    3. buildTypes
    4. productFlavors

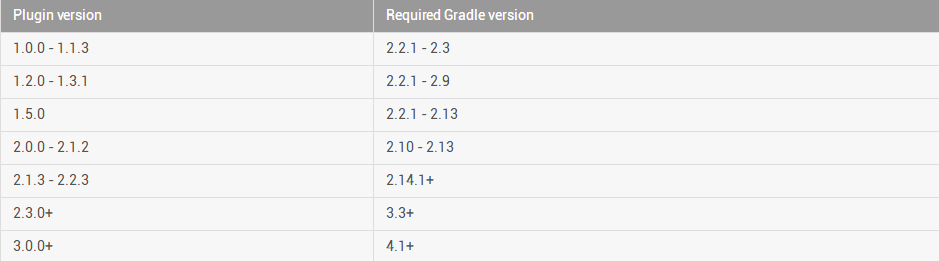
# Dependencies

# Apply plugin “...”

# Gradle和Gradle plugin的版本

对应关系如下图所示，（资料链接-[网址](https://developer.android.com/studio/releases/gradle-plugin.html" \l "updating-gradle)）：

* 1. Gradle的版本在app\gradle\wrapper\gradle-wrapper.properties
  2. Gradle plugin在app\build.gradle\buildscript\dependencies\classpath “com.android.tools.build:gradle:XXXXX”.
  3. GradleWrapper路径： C:\Users\Administrator\.gradle\wrapper\dists\gradle-3.3-all\55gk2rcmfc6p2dg9u9ohc3hw9中对应的gradle-3.3-all.zip



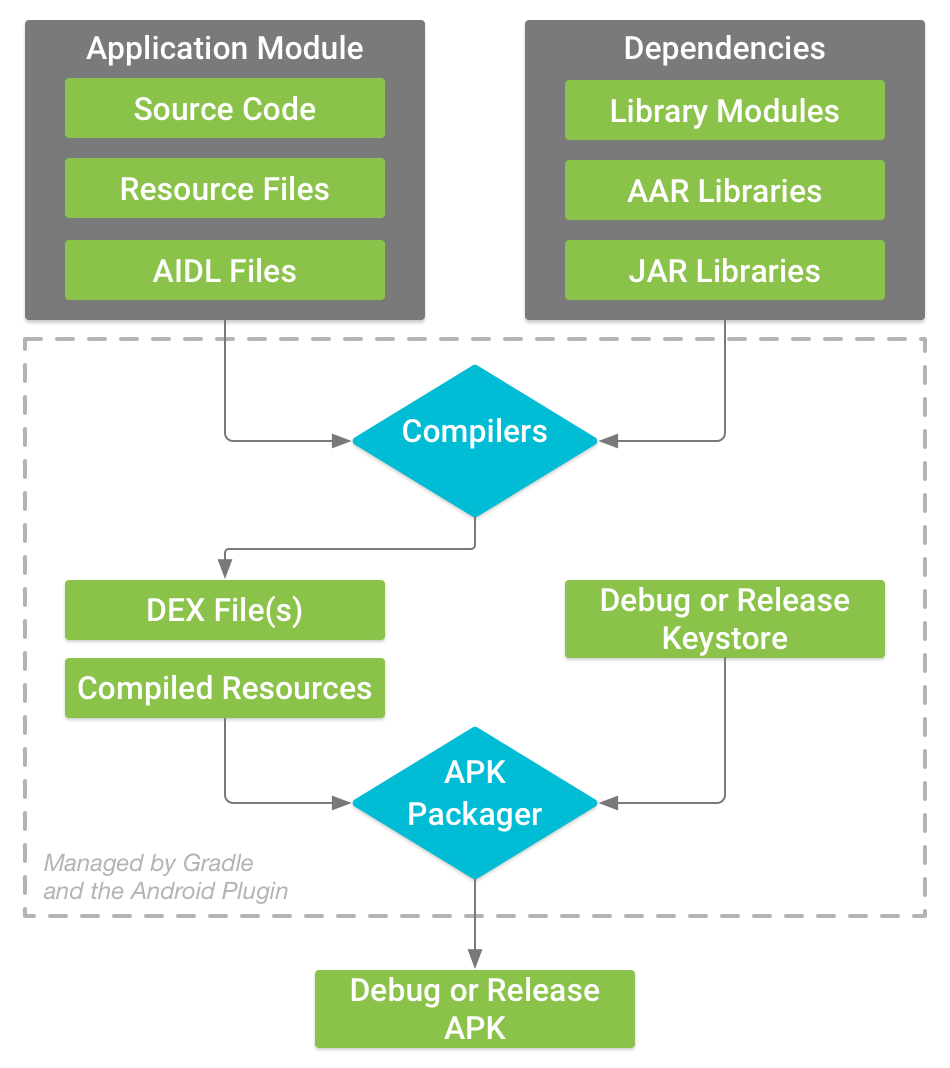
如何才能解决问题：build.gradle:classpath **'com.android.tools.build:gradle:2.2.0'**

**distributionUrl**=**https\://services.gradle.org/distributions/gradle-3.3-all.zip**

1. 关于[Instant Run](http://blog.csdn.net/guolin_blog/article/details/51271369)

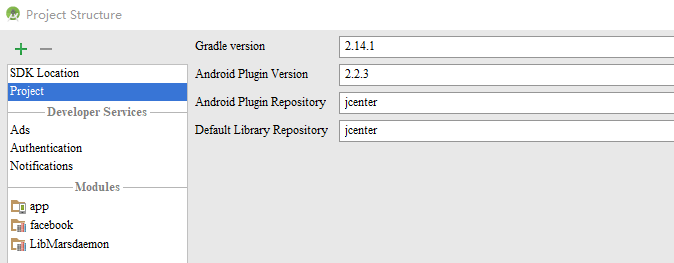
传统情况下，我们修改程序后重新运行一次程序需要经历 代码重新编译 -> 停止程序 -> 重新安装 -> 重新启动 这样一个过程，而Instant Run则尝试只将程序变更的部分部署到手机上，尽量避免重新安装或重新启动程序，以此大大提升调试程序的效率。

1. 从命令行构建应用。参考网址：[地址](https://developer.android.com/studio/build/building-cmdline.html)。
2. 编译链接过程如下图所示。

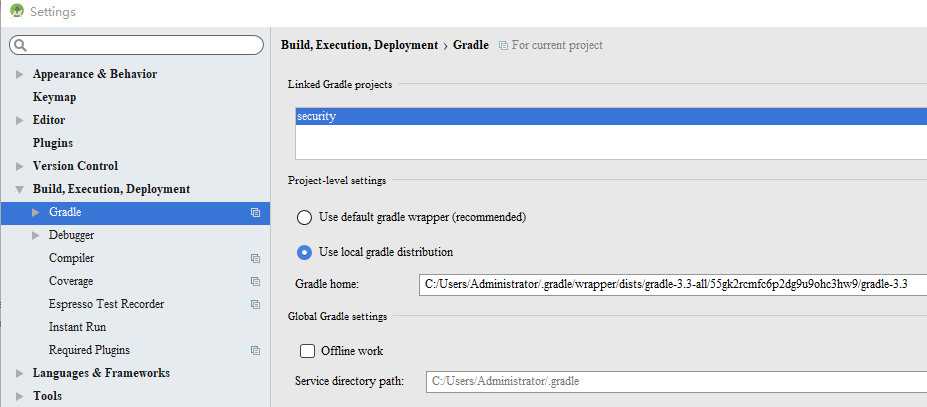


# 编译Error与解决办法

1. AS版本2.3 Beata 2，升级到“Instant Run”，提示错误：“The android plugin version 2.3.0-beta2 is too old, please update to the latest version. To override this check from the command line please set the ANDROID\_DAILY\_OVERRIDE environment variable to “...””
   1. 环境：Project配置：Gradle version 3.3; Android Plugin Version 2.3.0-beta2; Android Plugin Repository: jcenter; Default Library Repository: jcenter.
   2. 解决步骤：
2. Failed to open zip file. Gradle’s dependency cache may be corrupt (this sometimes occurs after a network connection timeout.) re-download dependencies and sync project(requires network)
   1. 环境：截图如下：



* 1. 解决步骤：打开如下对话框，选择对应本地的gradle文件即可。疑问：指定的gradle和配置的gradle版本不一致也能解决问题？



# 专题之AS编译慢的问题

当项目中引用的第三方模块太多，打开Android Studio的时候就会非常慢，提示一直在解决模块依赖，点击sync按钮同步第三方代码的时候也是巨慢，慢的原因是

1. 在gradle中引用第三方模块时采用maven方式依赖，每次打开Android Studio或者点击sync按钮时，都会去maven中央仓库去取第三方的库文件,一般是jar或者aar文件。
2. 如果本地没有该文件，则下载回来，由于通过网络访问Android Studio默认的maven中央仓库jcenter很慢，因此就导致同步会非常耗时，尤其是第一次同步，或者clean后重新build的时候，一般都会在十分钟左右。
3. 即使库文件已经下载到本地了，也会去访问maven仓库的，因此当过多的通过maven方式依赖第三方库的时候Android Studio就会非常慢。
4. Android Studio在编译的时候，如果没有同步第三方代码，或者任何一个gradle文件发生变化后，没有去同步，那么点击运行按钮会自动先同步，也就是先同步后编译，平时我们感觉编译慢的原因主要是同步代码太慢，真正同步完成后编译其实是很快的。
5. svn用来管理文本文件，但它不擅长管理二进制文件，这个时候maven就登场了，简单点说，maven就是管理二进制文件的svn，这样说应该可以理解了吧。
6. 同步代码时，Android Studio就会自动去下载作者发布的jar(或者aar)文件，如果作者发布了新的版本，则只需要简单的修改一下后面的版本号即可，不要这个库，直接删掉这行代码即可。
7. 当第一次同步代码后，第三方库都下载到本地了，将gradle设置为offline模式，这样下次就不必访问网络了，这样同步速度从几分钟缩短到几秒钟，但缺点是一旦引入新的maven依赖，本地不存在这个库文件，还是得去访问网络，同样同步慢。
8. 自己搭建maven仓库，将常用的第三方库放在自己搭建的maven服务器上，这样访问内部的maven服务器就快多了，又方便分模块开发和打包，但这样对服务器的管理麻烦，技术含量高，比较适合需要分模块开发的大型项目，而且当多个模块开发不同步，需要对模块分版本进行依赖、打包、发布的时候，必须采用这种方式，例如：模块A已经开发完毕，模块B正在开发，但这个时候要发版本了，怎么办呢？解决办法就是依赖模块B的上个版本，也就是上个版本的aar文件。
9. 将第三方库下载到本地，然后将代码复制到自己的项目中，这种方法是在代码级别上直接使用别人的代码，优点是再也不用依赖别人的库，项目模块少，库与自己项目融为一体，缺点是复制的时候太辛苦了，不仅要复制代码文件，还要复制资源文件，容易出错，另外耦合比较严重，删除这个库的时候也非常麻烦，同时别人的代码也破坏了自己的代码风格和命名规范。
10. 推荐的方法是：尽量使用第三方库的jar、aar文件导入自己的项目，或者将第三方库下载到本地，然后当做一个本地模块导入自己的项目，不要再使用gradle中的maven依赖了
11. 这样就不用苦逼的去复制代码，删除的时候也很方便,只需删除模块即可，实现模块即插即用，缺点是会导致项目的模块太多，不美观，以及很多eclipse写的库导入Android Studio后还要自己修改才能编译通过，很多库是Android工程，不是标准意义的Android library，好在现在很多有名的库都专门提供了library，直接导入即可，还有Android Studio已经可以直接将eclipse工程当做一个module导入了,所以这种方法还是很简单的这种方法也就是以前eclipse使用第三方库的方法。
12. 自己下载gradle到本地，然后在Android Studio指定gradle的位置，避免第一次创建项目时去下载gradle，那也是巨慢的一个过程。
13. 自己手动下载依赖库方法，[参考网址](http://blog.csdn.net/fuchaosz/article/details/51146091)。
14. 通过**命令行编译工程，**跟踪一下编译过程，就知道是在哪个步骤卡住了。
15. Gradlew的用处。
16. Instant Run.

# Aapt

1. 编译后，部署过程中遇到ProcessException: Failed to execute aapt的错误，就需要更改compileSdkVesion或者下载对应sdk的aapt.