

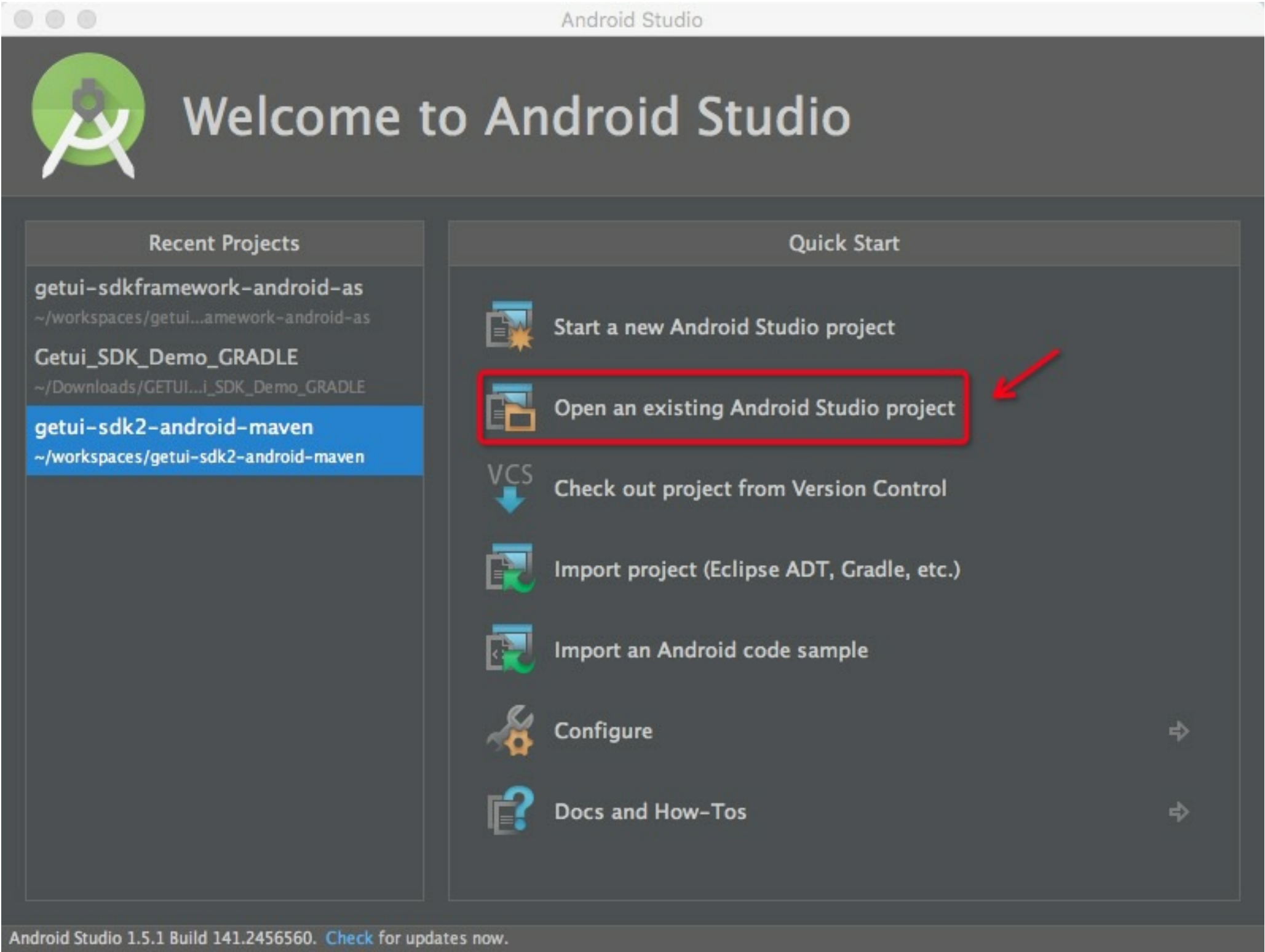
# Android Studio集成步骤

## 第一步： 在个推开发者平台创建应用

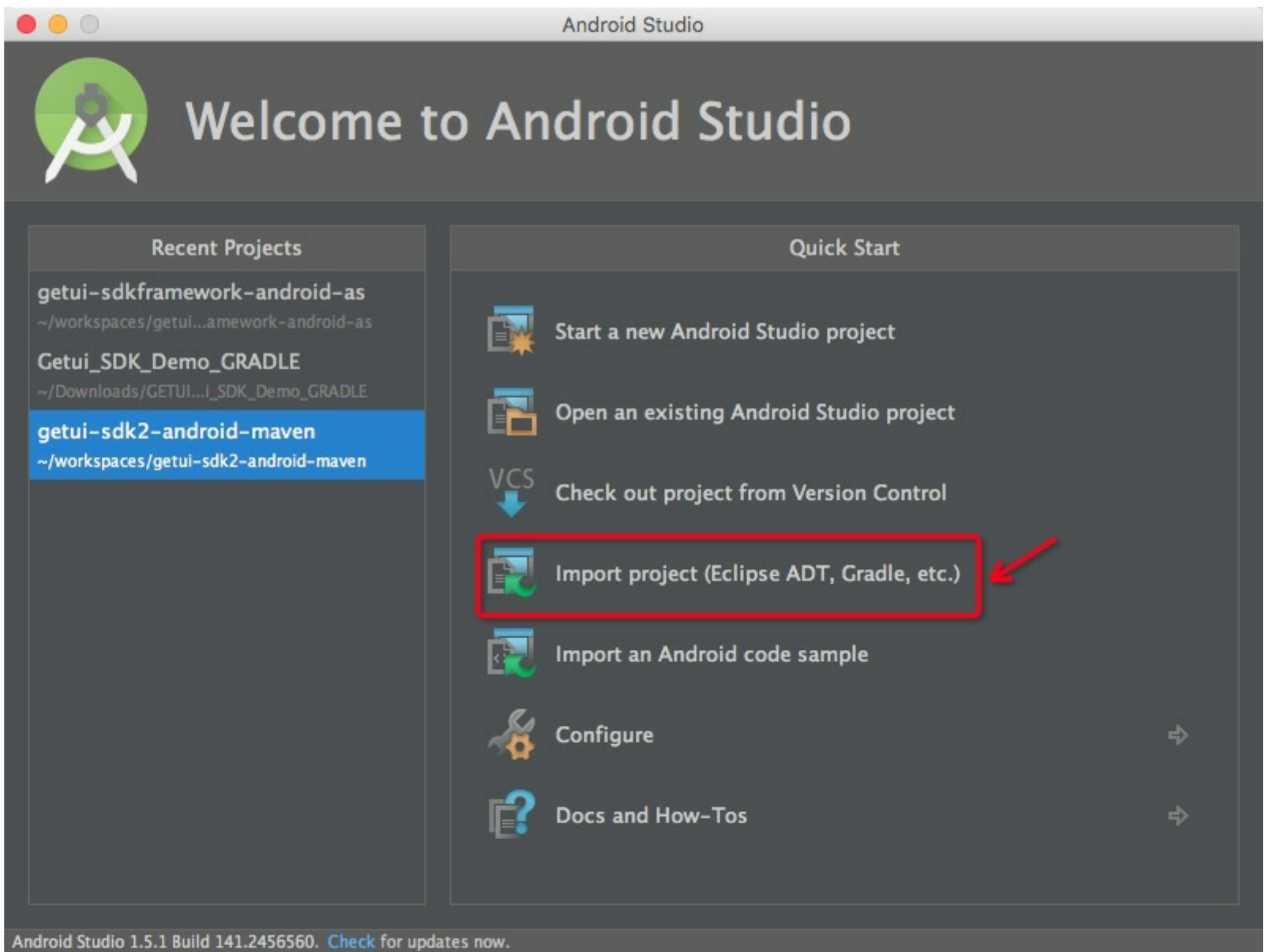
登录[个推开发者平台](#)，登记应用，并获取到相应的APPKEY、APPID、APP SECRET信息。参见 [登记应用](#)

## 第二步： 打开项目工程

启动Android Studio, 打开您之前创建的Android项目工程：



如果需要从原有的Eclipse项目导入， 请选择Import project (Eclipse ADT, Gradle, etc.)：

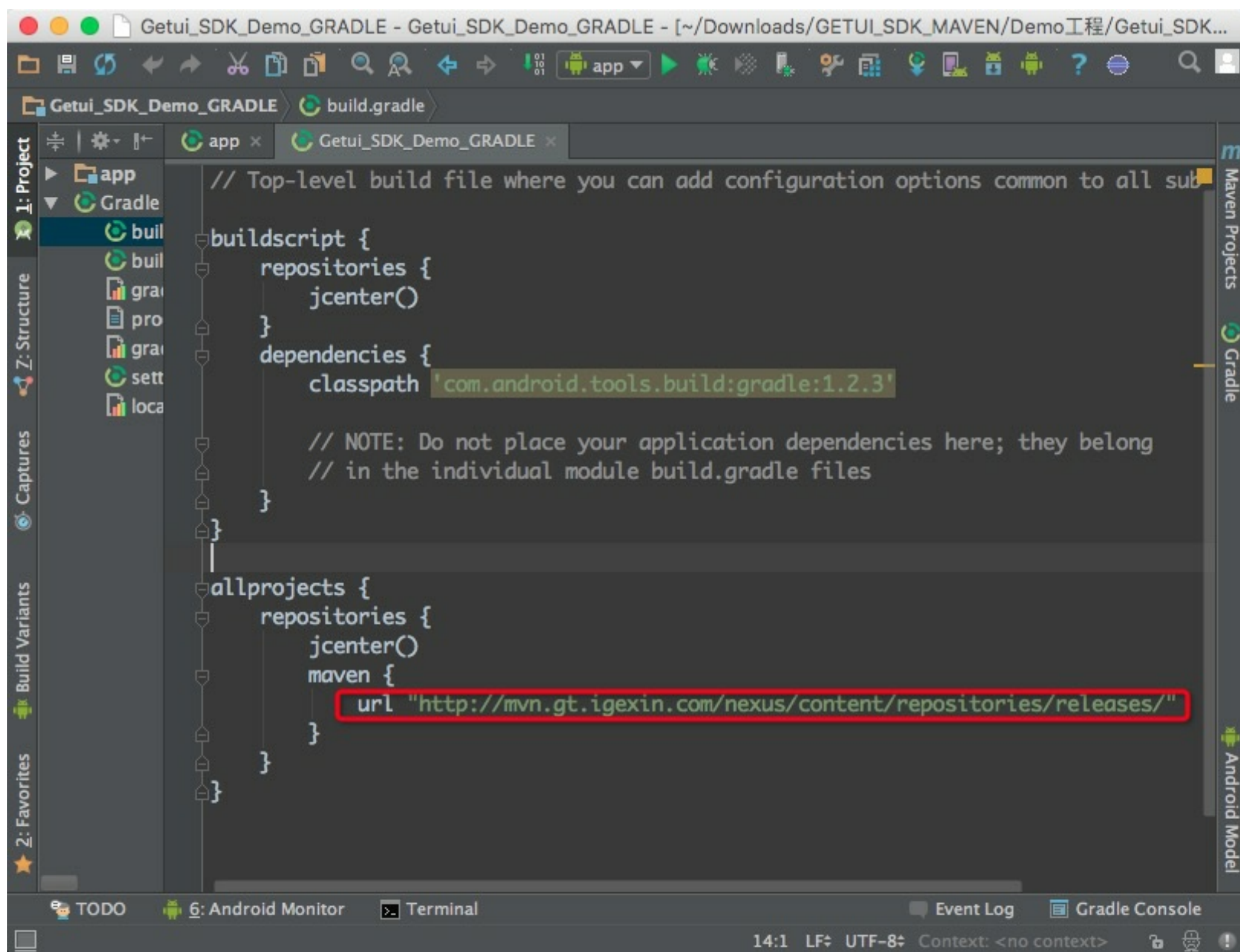


## 第三步：资源文件及配置引入

### 方法一：Maven自动导入

第一步：添加Maven库地址

在以项目名为命名的顶层build.gradle文件中，添加个推maven库地址，如下所示：

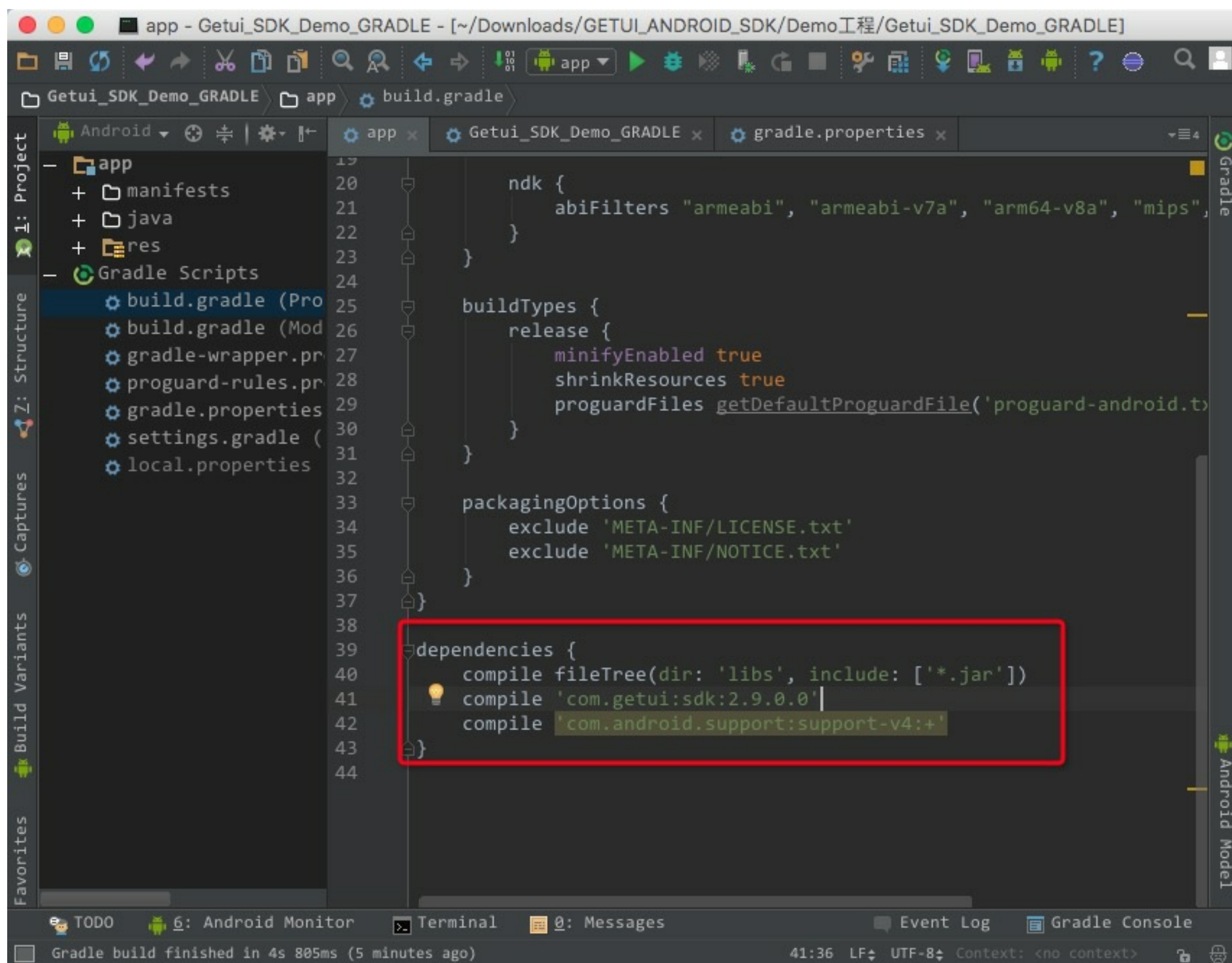


```
//Maven URL地址
maven {
    url "http://mvn.gt.igexin.com/nexus/content/repositories/releases/"
}
```

## 第二步：配置依赖

在app/build.gradle文件中配置依赖库，如下图所示：



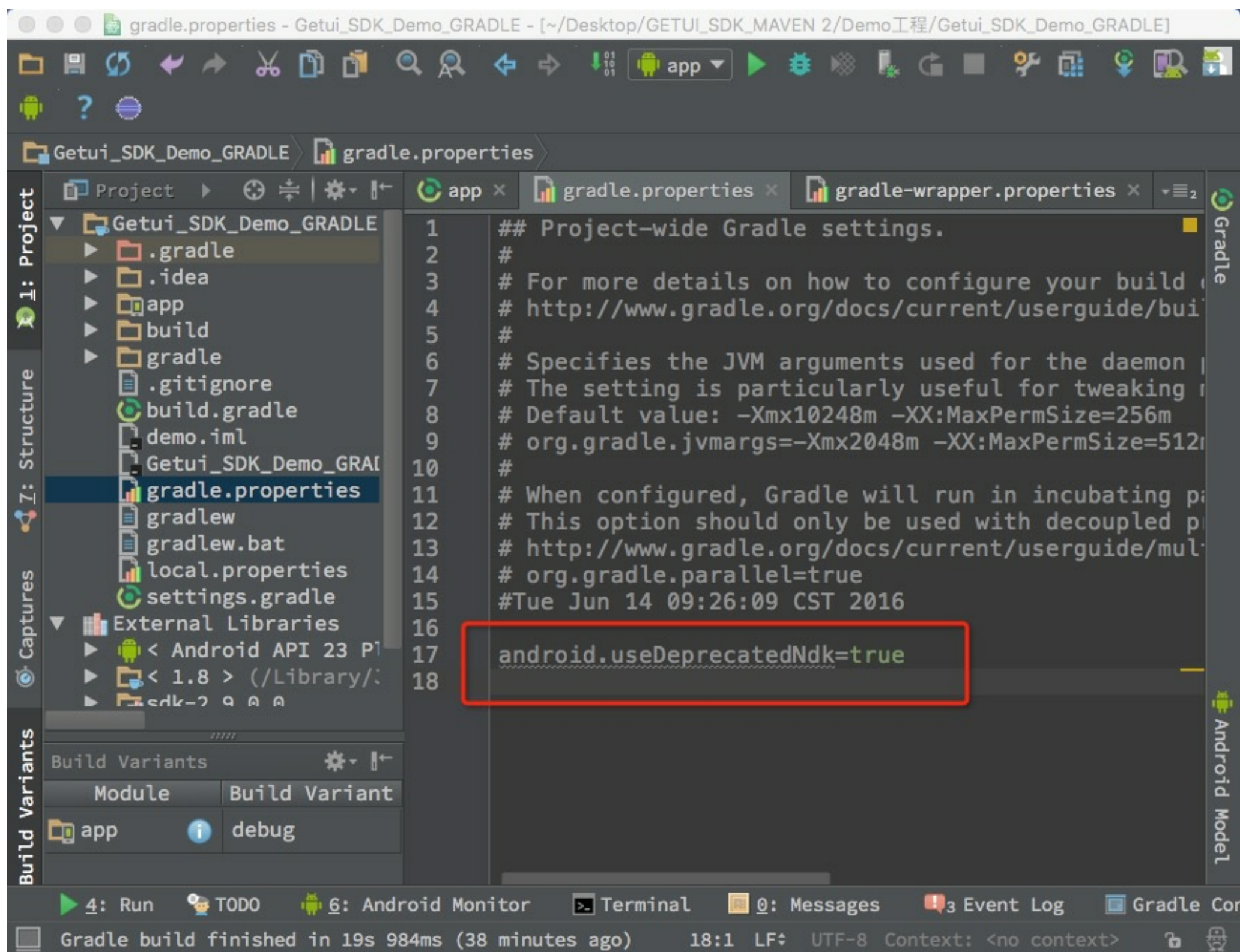


```
//相关配置
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.getui:sdk:2.9.0.0'
    compile 'com.android.support:support-v4:+'
}
```

注：开发者可以指定com.getui:sdk的版本号，也可以指定使用最新的sdk版本: compile 'com.getui:sdk:+'

第三步：配置 so 库

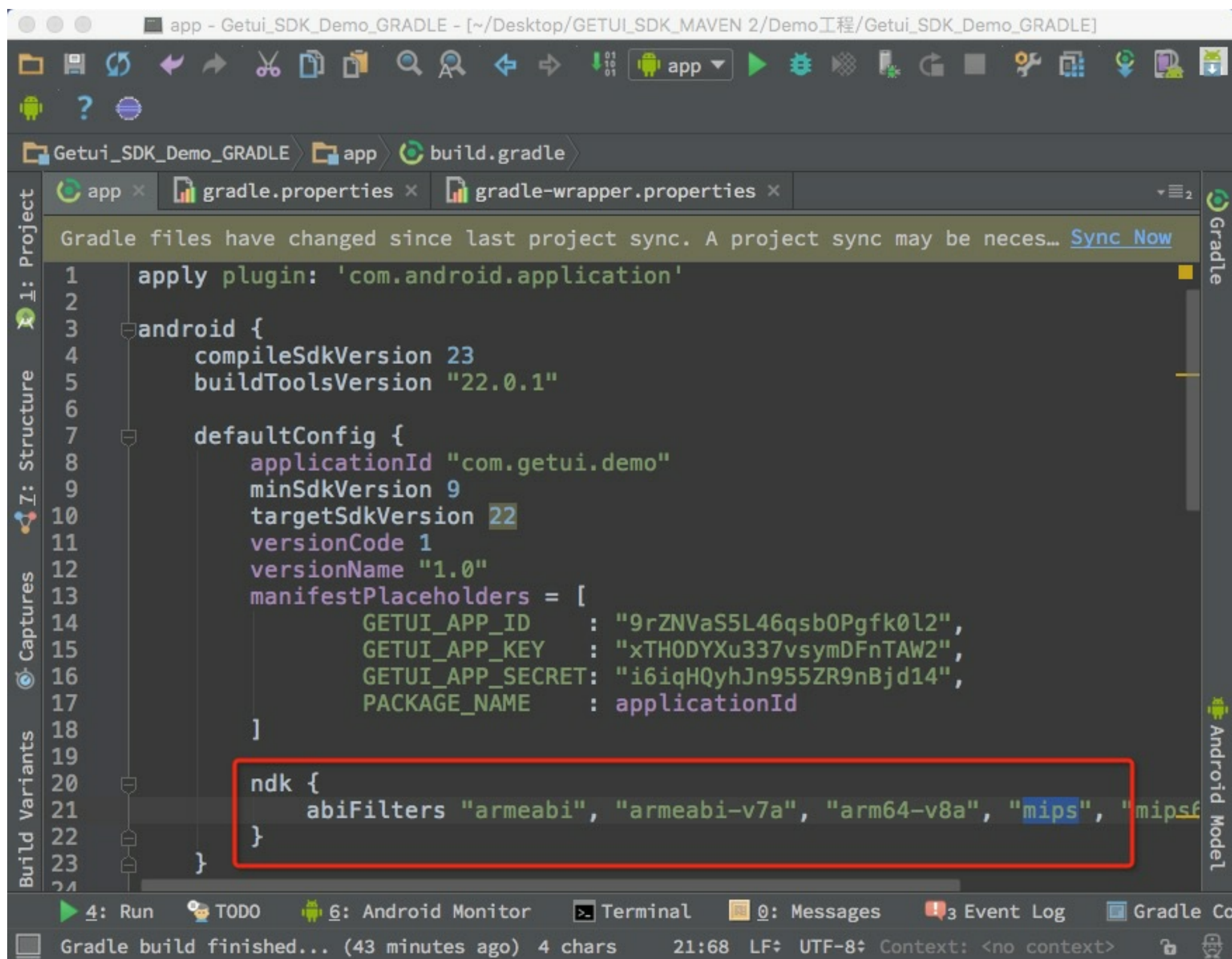
在gradle.properties文件中配置useDeprecatedNdk，如下图所示：



android.useDeprecatedNdk=true

在app/build.gradle文件中android/defaultConfig下指定需要 cup 架构的 so 库，如下图所示：





目前支持 armeabi、armeabi-v7a、arm64-v8a、mips、mips64、x86、x86\_64，根据需要指定即可

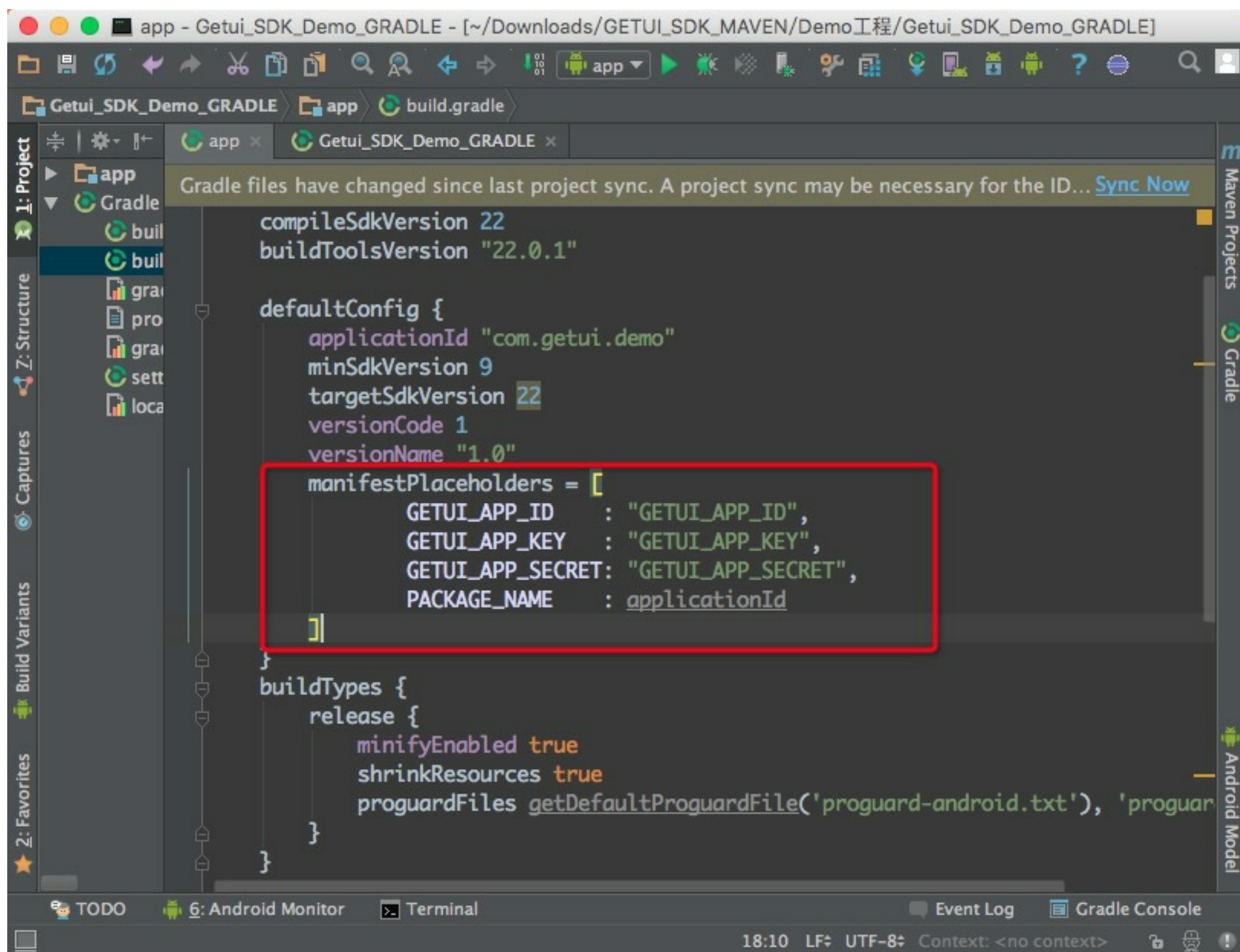
注：如果 project 中包含其他 so 库且只支持某几种 cpu 架构，那么应该根据其他 so 库支持的 cpu 架构来配置

```

android {
    ...
    defaultConfig {
        ...
        ndk {
            abiFilters "armeabi", "armeabi-v7a", "arm64-v8a", "mips", "mips64", "x86", "x86_64"
        }
    }
}

```

第四步：配置个推应用参数



//参数说明

```
manifestPlaceholders = [
    GETUI_APP_ID : "APP_ID",
    GETUI_APP_KEY : "APP_KEY",
    GETUI_APP_SECRET : "APPSECRET",
    PACKAGE_NAME : applicationId
]
```

//APP\_ID、APP\_KEY、APP\_SECRET请根据个推开发者后台申请到的应用参数进行相应替换

## 第五步：配置透传

根据业务需要，在AndroidManifest.xml添加用于接收透传消息的BroadcastReceiver，第三方开发者需要自行实现该BroadcastReceiver，以便接收CID信息和服务端推送的透传消息。

```
<!-- 配置第三方Receiver -->
<receiver
    <!-- 此处com.getui.demo.PushDemoReceiver，需要替换成开发者自己的BroadcastReceiver文件全名-->
    android:name="com.getui.demo.PushDemoReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="com.igexin.sdk.action.${GETUI_APP_ID}" />
    </intent-filter>
</receiver>
```



# 方法二：手动导入

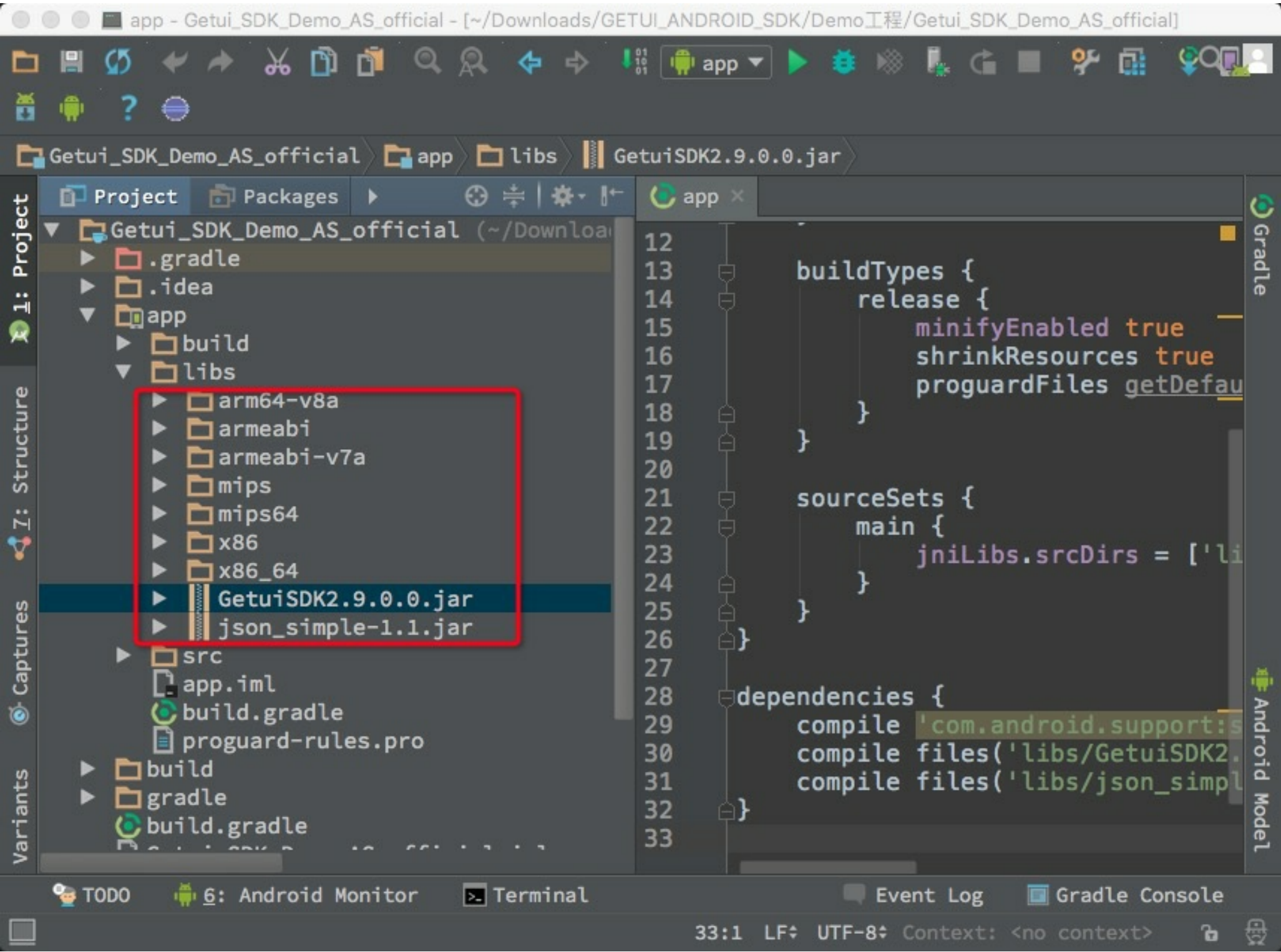
注：老版本升级2.9+

- 1.删除原来所有旧的布局文件，加上新的getui\_notification.xml
- 2.替换旧的GetuiSDKxxx.jar并删除GetuiExt.xxx.jar和所有对应目录下的libgetuiext.so
- 3.修改AndroidManifest.xml中个推广播和服务的配置，修改后见下方第三步

第一步：导入个推SDK

将SDK资料包中“GETUI\_ANDROID\_SDK\资源文件”目录下的GetuiSdk-xxx.jar、so文件夹子文件复制到app模块目录下的libs文件夹中。

注：导入需要的 cpu 架构的 so 库即可



打开app/build.gradle，在dependencies中添加相关jar包的引用：

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile files('libs/GetuiSDK2.9.0.0.jar')
    compile 'com.android.support:support-v4'
}
```

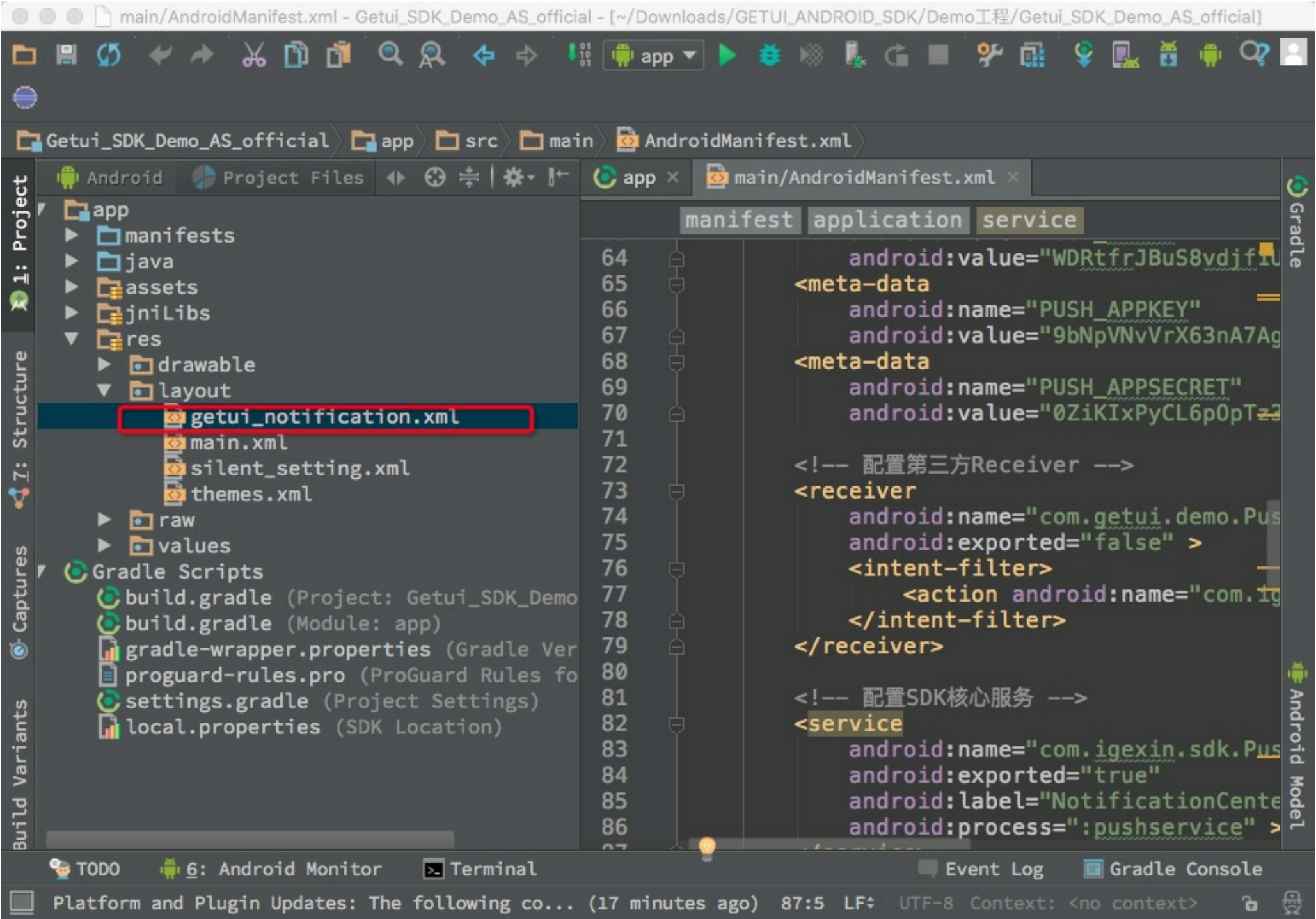
在app/build.gradle文件中的android{}中添加jnilib引用，代码如下：



```
sourceSets {
    main {
        jniLibs.srcDirs = ['libs']
    }
}
```

## 第二步：导入布局文件

将“GETUI\_ANDROID\_SDK\资源文件\layout”下的xml布局文件复制到app模块的layout文件夹中：



注：为了支持最新的展开式样式和浮动通知，必须导入getui\_notification.xml布局文件

### 第三步：添加服务声明

在Application标签内加入如下服务声明：

```
<!-- 个推SDK配置开始 -->
<!-- 配置的第三方参数属性 -->
<meta-data
    android:name="PUSH_APPID"
    android:value="你的APPID" /> <!-- 替换为第三方应用的APPID -->
<meta-data
    android:name="PUSH_APPKEY"
    android:value="你的APPKEY" /> <!-- 替换为第三方应用的APPKEY -->
<meta-data
    android:name="PUSH_APPSECRET"
```

```
        android:value="你的APPSECRET" /> <!-- 替换为第三方应用的APPSECRET -->
<!-- 配置SDK核心服务 -->
<service
    android:name="com.igexin.sdk.PushService"
    android:exported="true"
    android:label="NotificationCenter"
    android:process=":pushservice" >
    <intent-filter>
        <action android:name="com.igexin.sdk.action.service.message"/>
    </intent-filter>
</service>

<service
    android:name="com.igexin.sdk.PushServiceUser"
    android:exported="true"
    android:label="NotificationCenterUser">
    <intent-filter>
        <action android:name="com.igexin.sdk.action.user.message"/>
    </intent-filter>
</service>

<receiver android:name="com.igexin.sdk.PushReceiver" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
        <action android:name="android.intent.action.USER_PRESENT" />
        <action android:name="com.igexin.sdk.action.refreshls" />
        <!-- 以下三项为可选的action声明，可大大提高service存活率和消息到达速度 -->
        <action android:name="android.intent.action.MEDIA_MOUNTED" />
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
    </intent-filter>
</receiver>
<receiver
    android:name="com.igexin.sdk.PushManagerReceiver"
    android:exported="false" >
    <intent-filter>
        <action android:name="com.igexin.sdk.action.pushmanager" />
    </intent-filter>
</receiver>
<activity
    android:name="com.igexin.sdk.PushActivity"
    android:excludeFromRecents="true"
    android:exported="false"
    android:process=":pushservice"
    android:taskAffinity="com.igexin.sdk.PushActivityTask"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
<activity
    android:name="com.igexin.sdk.GActivity"
    android:excludeFromRecents="true"
    android:exported="true"
    android:process=":pushservice"
    android:taskAffinity="com.igexin.sdk.PushActivityTask"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
<service
    android:name="com.igexin.download.DownloadService"
```



```
        android:process=":pushservice" />
<receiver android:name="com.igexin.download.DownloadReceiver" >
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
    </intent-filter>
</receiver>
<provider
    android:name="com.igexin.download.DownloadProvider"
    <!-- 把"你的包名"替换为第三方应用的包名 -->
    android:authorities="downloads.你的包名"
    android:exported="true"
    android:process=":pushservice" />
<!-- 个推SDK配置结束 -->
```

注：需要将注释部分参数替换为个推开发者平台上应用所分配到的参数

#### 第四步：添加权限声明

在Application标签外加入个推SDK运行时需要的权限：

```
<!-- 解决Android L上通知显示异常问题，targetSdkVersion需要设置成22 -->
<uses-sdk
    android:minSdkVersion="9"
    android:targetSdkVersion="22" />
<!-- 个推SDK权限配置开始 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.GET_TASKS" />

<!-- iBeacon功能与个推3.0电子围栏功能所需要的权限为非必需的可选择权限，可以选择性配置，以便使用个推3.0电子围栏功能 -->
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<!-- 个推3.0电子围栏功能所需权限 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<!-- 浮动通知权限 -->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<!-- 自定义权限 -->
<uses-permission android:name="getui.permission.GetuiService.你的包名" />
<!-- 替换为第三方应用的包名 -->
<permission
    android:name="getui.permission.GetuiService.你的包名"
    android:protectionLevel="normal" >
</permission><!-- 替换为第三方应用的包名 -->
<!-- 个推SDK权限配置结束 -->
```

自定义权限解释：部分手机型号不能正常运行个推SDK，需添加自定义权限进行配置。

## 第五步：完整示例

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.getui.demo"
    android:versionCode="2"
    android:versionName="2.0.0">
    <!-- 解决Android L上通知显示异常问题，targetSdkVersion需要设置成22 -->
    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="22"/>
    <!-- 个推SDK权限配置开始 -->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.GET_TASKS"/>

    <!-- iBeacon功能所需权限 -->;
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <!-- 个推3.0电子围栏功能所需权限 -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

    <!-- 浮动通知权限 -->
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
    <!-- 自定义权限 -->
    <uses-permission android:name="getui.permission.GetuiService.com.getui.demo"/>
    <permission
        android:name="getui.permission.GetuiService.com.getui.demo"
        android:protectionLevel="normal">
    </permission>
    <!-- 个推SDK权限配置结束 -->
    <application
        android:icon="@drawable/demo"
        android:label="@string/app_name"
        android:persistent="true">
        <!-- 第三方应用配置 -->
        <activity
            android:name=".GetuiSdkDemoActivity"
            android:label="@string/app_name"
            android:launchMode="singleTop">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
```



```
<!-- 在上面加入你的你的activity配置 -->
<!-- 个推SDK配置开始 -->
<!-- 配置的第三方参数属性 -->
<meta-data
    android:name="PUSH_APPID"
    android:value="wT1tGnaFC98Kgpfoi2u7g6"/>
<meta-data
    android:name="PUSH_APPKEY"
    android:value="wT1tGnaFC98Kgpfoi2u7g6"/>
<meta-data
    android:name="PUSH_APPSECRET"
    android:value="my9R0U9s2Y8vLSfeToj6N5"/>
<!-- 配置第三方Receiver -->
<receiver
    android:name="com.getui.demo.PushDemoReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="com.igexin.sdk.action.wT1tGnaFC98Kgpfoi2u7g6"/>
    </intent-filter>
</receiver>
<!-- 配置SDK核心服务 -->
<service
    android:name="com.igexin.sdk.PushService"
    android:exported="true"
    android:label="NotificationCenter"
    android:process=":pushservice">
    <intent-filter>
        <action android:name="com.igexin.sdk.action.service.message"/>
    </intent-filter>
</service>

    <service
        android:name="com.igexin.sdk.PushServiceUser"
        android:exported="true"
        android:label="NotificationCenterUser">
        <intent-filter>
            <action android:name="com.igexin.sdk.action.user.message"/>
        </intent-filter>
    </service>

<receiver android:name="com.igexin.sdk.PushReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
        <action android:name="android.intent.action.USER_PRESENT"/>
        <action android:name="com.igexin.sdk.action.refreshls"/>
        <!-- 以下三项为可选的action声明，可大大提高service存活率和消息到达速度 -->
        <action android:name="android.intent.action.MEDIA_MOUNTED"/>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
    </intent-filter>
</receiver>
<receiver
    android:name="com.igexin.sdk.PushManagerReceiver"
    android:exported="false">
    <intent-filter>
```

```

        <action android:name="com.igexin.sdk.action.pushmanager"/>
    </intent-filter>
</receiver>
<activity
    android:name="com.igexin.sdk.PushActivity"
    android:excludeFromRecents="true"
    android:exported="false"
    android:process=":pushservice"
    android:taskAffinity="com.igexin.sdk.PushActivityTask"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
<activity
    android:name="com.igexin.sdk.GActivity"
    android:excludeFromRecents="true"
    android:exported="true"
    android:process=":pushservice"
    android:taskAffinity="com.igexin.sdk.PushActivityTask"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
<service
    android:name="com.igexin.download.DownloadService"
    android:process=":pushservice"/>
<receiver android:name="com.igexin.download.DownloadReceiver">
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
    </intent-filter>
</receiver>
<provider
    android:name="com.igexin.download.DownloadProvider"
    android:authorities="downloads.com.getui.demo"
    android:exported="true"
    android:process=":pushservice"/>
<!-- 个推SDK配置结束 -->
</application>
</manifest>

```

## 第六步：配置透传

根据业务需要，在AndroidManifest.xml添加用于接收透传消息的BroadcastReceiver，第三方开发者需要自行实现该BroadcastReceiver，以便接收CID信息和服务端推送的透传消息。

```

<!-- 配置第三方Receiver -->
<receiver
    <!-- 此处com.getui.demo.PushDemoReceiver，需要替换成开发者自己的BroadcastReceiver -->
    android:name="com.getui.demo.PushDemoReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="com.igexin.sdk.action.你的APP_ID" />
    </intent-filter>
</receiver>

```

**注：需要替换APP\_ID参数**

## 第七步：配置混淆

在混淆文件中加入如下配置即可：



```
-dontwarn com.igexin.**
-keep class com.igexin.**{*;}

```

## 第四步：配置可选权限

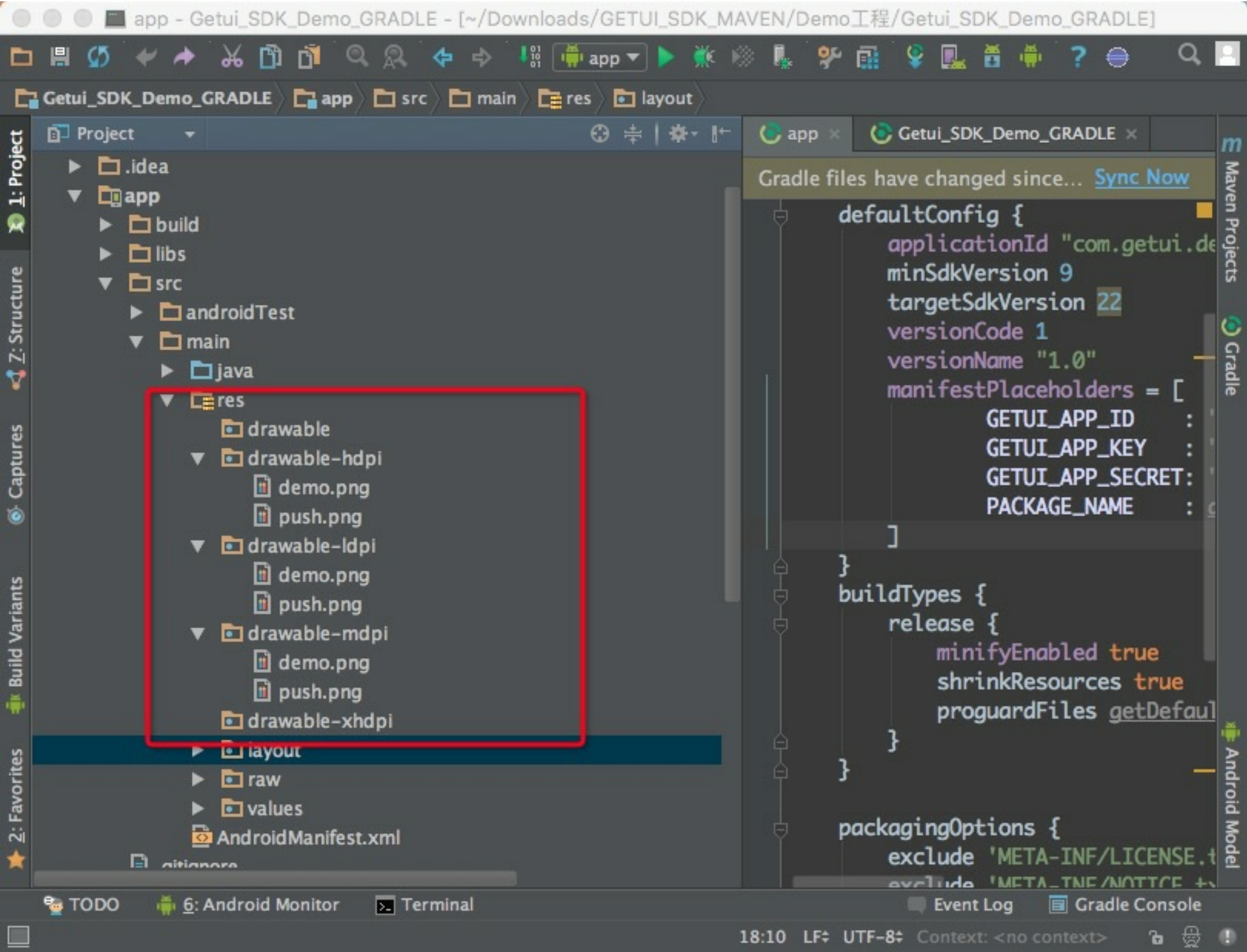
该接入方式已包含个推服务所需必备权限，在此之外，您还可以在自己的AndroidManifest.xml中配置以下可选权限，以便使用个推3.0电子围栏功能。

```
<!-- iBeacon功能所需权限 -->;
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<!-- 个推3.0电子围栏功能所需权限 -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

```

## 第五步：导入通知栏图标

为了修改通知栏提示图标，请在res/drawable-hdpi/、res/drawable-mdpi/、res/drawable-ldpi/等各分辨率资源目录下，放置相应尺寸的push.png图片。



该图标将会作为通知图标展示在通知栏顶部，如下所示：



## 第六步：初始化SDK

在您应用程序主Activity里导入PushManager类，如下所示：

```
import com.igexin.sdk.PushManager;
```

然后在您应用程序启动初始化阶段，初始化SDK：

```
PushManager.getInstance().initialize(this.getApplicationContext());
```

注：该方法必须在Activity或服务类内调用，一般情况下，可以在Activity的onCreate()方法中调用。由于应用每启动一个新的进程，就会调用一次Application的onCreate()方法，而个推SDK是一个独立的进程，因此如果在Application的onCreate()中调用initialize接口，会导致SDK初始化在一个应用中多次调用，所以不建议在Application继承类中调用个推SDK初始化接口。

建议应用程序每次启动时都调用一次该初始化接口。

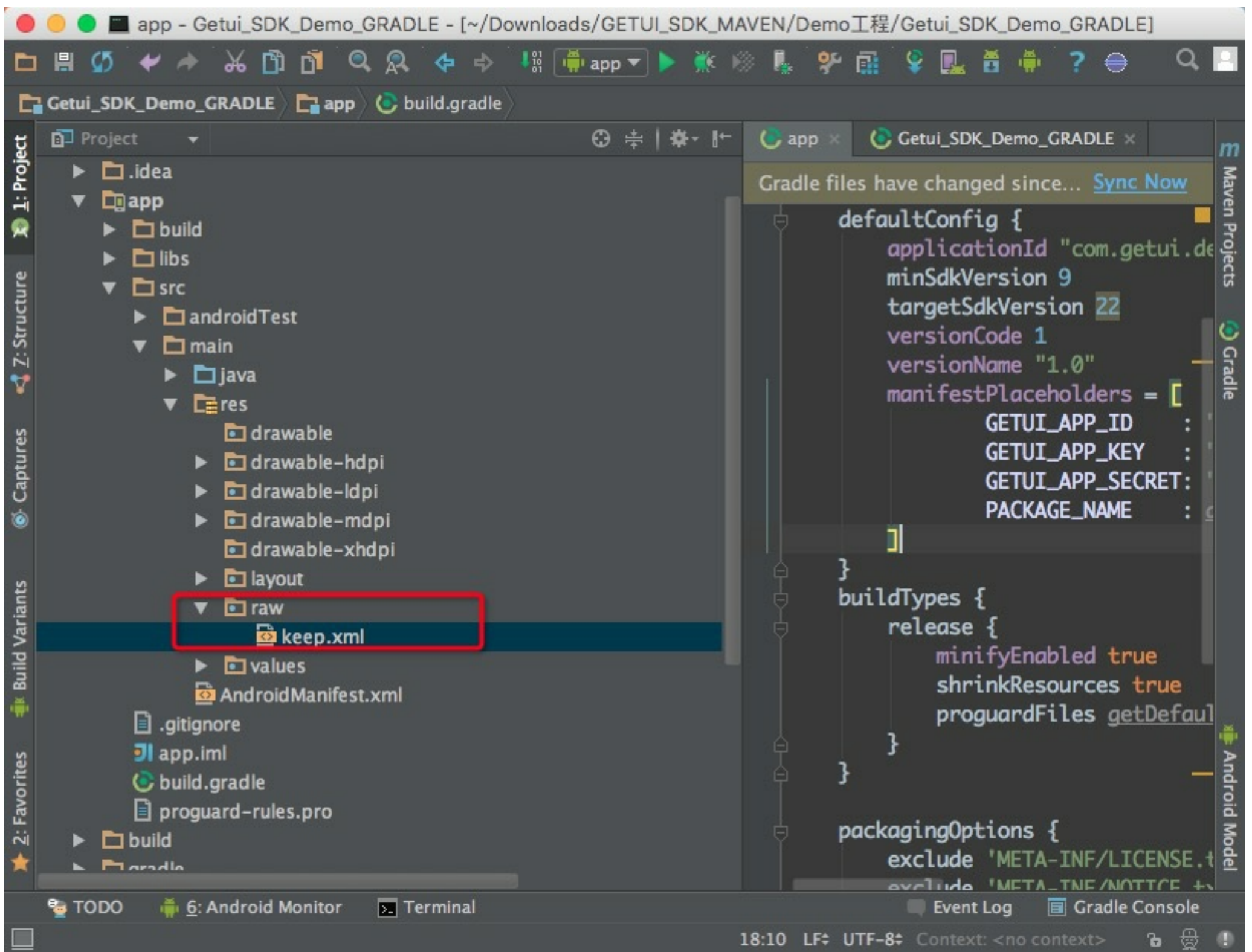
## 第七步：资源精简配置

如果您的工程启用了资源精简，即在build.gradle中指定如下参数：

```
buildTypes {
    release {
        minifyEnabled true
        shrinkResources true
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
}
```

需要在res/raw中添加keep.xml，明确指定个推SDK所需的layout资源文件不能被精简，keep.xml文件：





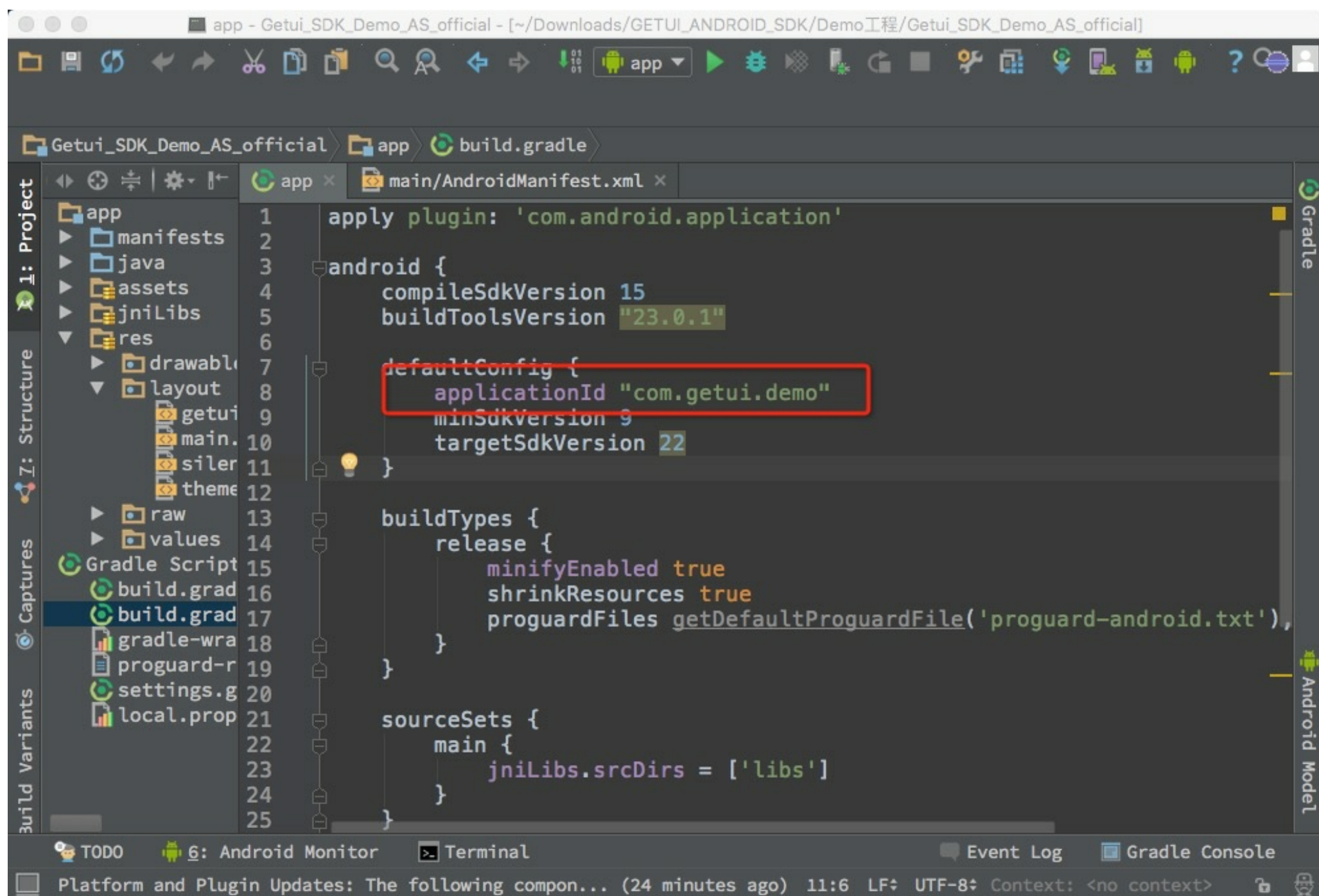
keep.xml文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources
    xmlns:tools="http://schemas.android.com/tools"
    tools:keep="@layout/getui_notification"/>
```

如此可以完成layout资源保护工作。

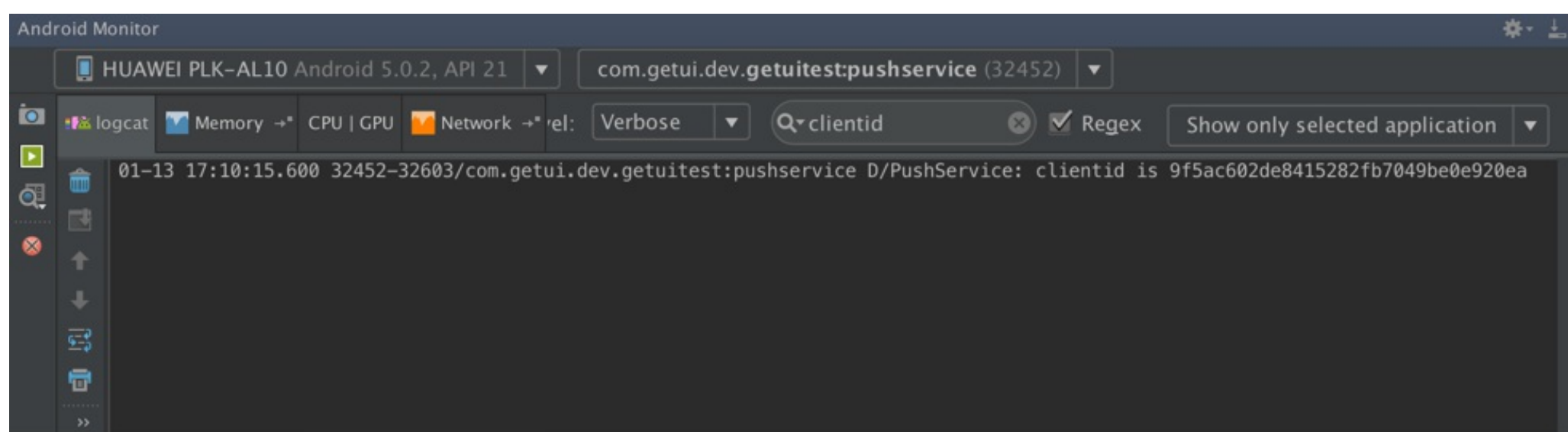
## 第八步：确认gradle配置

确认gradle文件中的applicationId和参数中需要的包名一致，如图：



## 第九步：测试

在手机或模拟器上运行您的工程，查看Android Monitor信息，如图所示。在搜索框中输入“clientid”可以看到“clientid is xxx”，则意味则初始化SDK成功，并获取到相应的cid信息，恭喜你:-D，可以开始进行推送测试了。



登录 [个推开发者平台](#)，点击应用管理，进入待测试应用的推送界面：





首页

用户分组

自定义事件

数据总览

VIP

PushGeTui

[退出]



创建推送

推送通知

透传消息

分组对比测试



推送记录

推送数据

推送统计

用户数据



应用配置

别名管理

应用标签

故障排查

推送通知 ?

展示内容：

默认

安卓

图片

\* 通知标题：

请输入通知标题

0/20

\* 通知内容：

请输入通知内容

0/50

\* 展开式通知：

禁用

文本

大图

\* 目标平台：

Android

\* 目标用户：

全部用户

标签用户

特定用户

用户分组

\* 后续动作：

启动应用

打开链接

下载应用

\* 推送方式：

按时间

即时

定时

是否进离线消息：

是

否

\* 有效时长：

2

小时, 该时间段内cid在线过的用户均可收到通知。(0- 72 小时内的正整数)

定时展示：

否

是

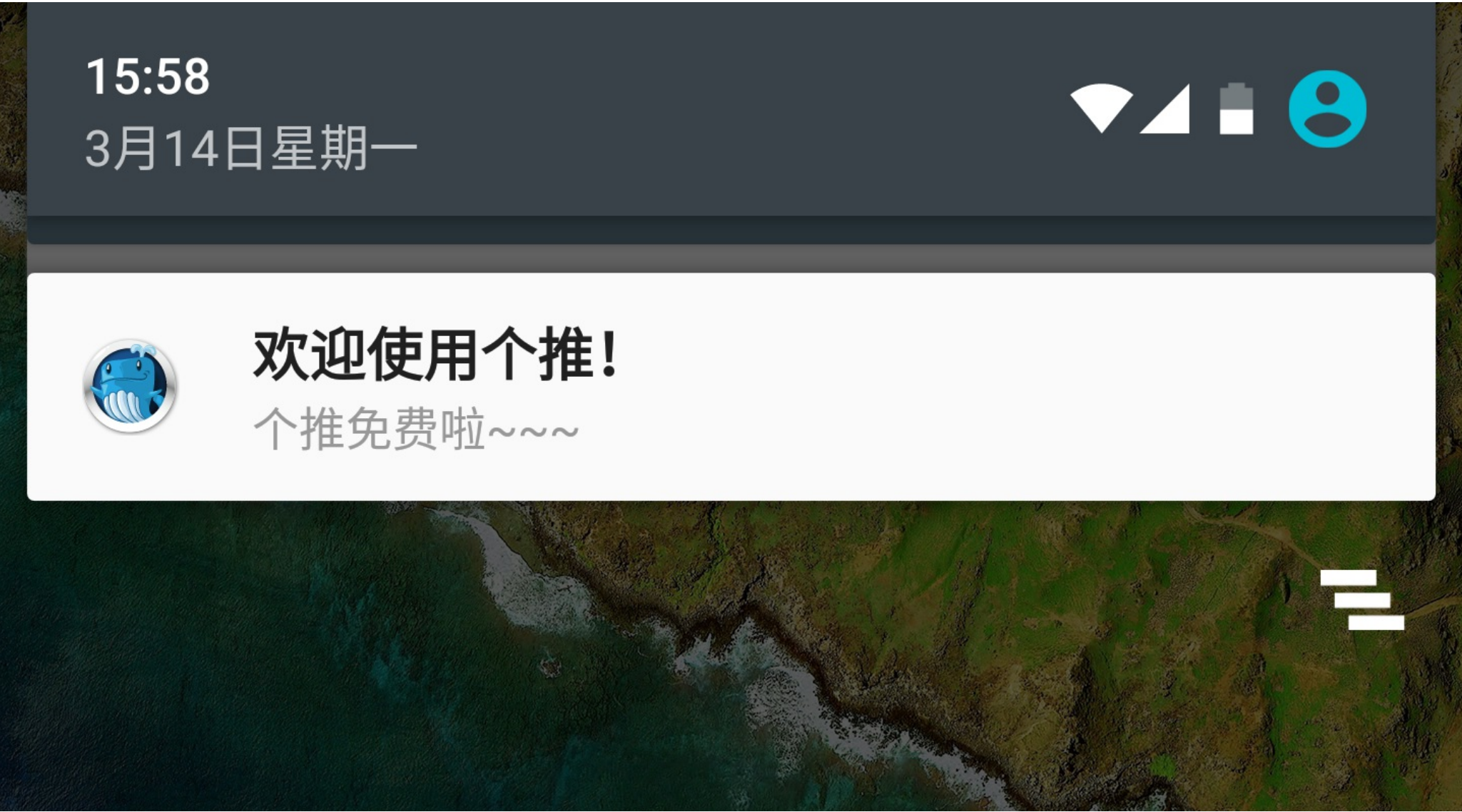
高级设置

发送



依填写相应的通知标题、通知内容，进行通知推送，具体推送方法见：[创建推送通知](#)

如果手机或模拟器收到通知，如图所示：恭喜您，SDK接入已经成功！

A screenshot of an Android emulator interface. At the top, the status bar shows the time 15:58, the date 3月14日星期一, and icons for Wi-Fi, signal strength, battery, and a user profile. Below the status bar, a white notification banner is displayed. On the left of the banner is the '个推' (GeTui) logo, a blue whale-like creature. To the right of the logo, the text reads '欢迎使用个推!' (Welcome to use GeTui!) in large black font, followed by '个推免费啦~~~' (GeTui is free now~~~) in a smaller grey font. In the bottom right corner of the emulator screen, there are three horizontal white lines, likely representing a menu or navigation bar.





Google



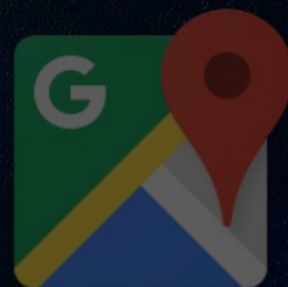
Play 商店



个推演示



Android Pay



地图

