

Android Studio快速集成（推荐）

关于Android Studio

Android Studio是Google力推的Android开发环境，在IntelliJ IDEA基础上进行了大量功能完善和优化，包括：

- 基于Gradle的构建支持
- Android 专属的重构工具和Instant Run快速修复技术
- 功能强大的布局编辑器，可以让你拖拉 UI 控件并进行效果预览
- 全新的 Android 模拟器大约比之前的模拟器快 3 倍，同时由于 ADB 的增强，传输应用和数据到模拟器上的速度比到物理设备上快 10 倍。
- 提供性能分析工具以捕获性能、可用性、版本兼容性问题

因此我们强烈推荐Android开发者将现有项目迁移到Android Studio环境，并在Android Studio下更快地实现个推SDK的集成工作。

前言

- 本文档介绍Android Studio提供的基于Maven的快速集成方案，配置简单、不容易出问题、后续更新维护方便，因此我们强烈推荐应用开发者根据本文档步骤进行个推集成。
- 本文档适用SDK版本：2.9.5.0及以后
- 请参考 `Getui_SDK_Demo_AS_maven` Demo工程

1. 创建个推应用

- 请登录 <http://dev.getui.com>，选择 `登记应用` 并填写应用名称和包名信息，完成应用创建：



登记应用 ?

* 应用名称 :

推送测试

4 / 30

* 应用类型 :

其他

* 应用平台 :

 Android ☒

 iOS ☐

* 应用标识 :

com.example.pushtest ?

登记

取消

- 点击 应用配置 ，获取到相应的 AppID 、 AppKey 、 AppSecret 信息：

testapp

创建推送

推送通知

透传消息

分组对比测试

数据统计

推送记录

推送数据

推送统计

用户数据

配置管理

应用配置

别名管理

应用标签

故障排查

应用配置 ?

删除应用

应用图标 :



25K大小以内

上传logo

应用名称 :

testapp

应用类型 :

其他

AppID: ?

3Y6XCpc9jp5HD7cxJDh0R9

AppSecret : ?

zK5wneLOqE6DZKPsB58FQ3


AppKey : ?

eea5EHdvem6ZcMMTKaX167

MasterSecret : ?

FRwdmp024a6l8ahef38O94

应用平台 :

 Android ☒

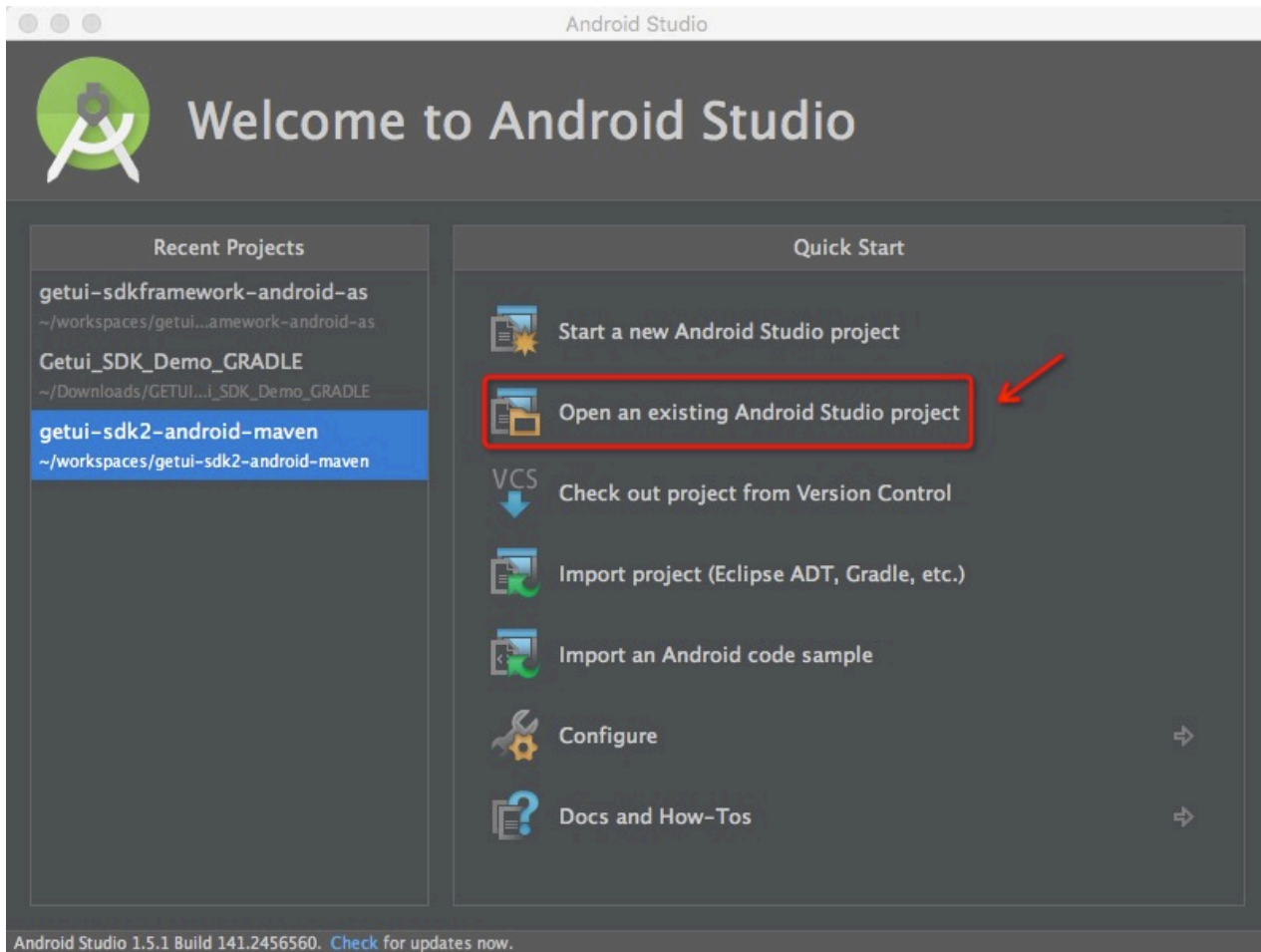
 iOS ☐

应用标识 :

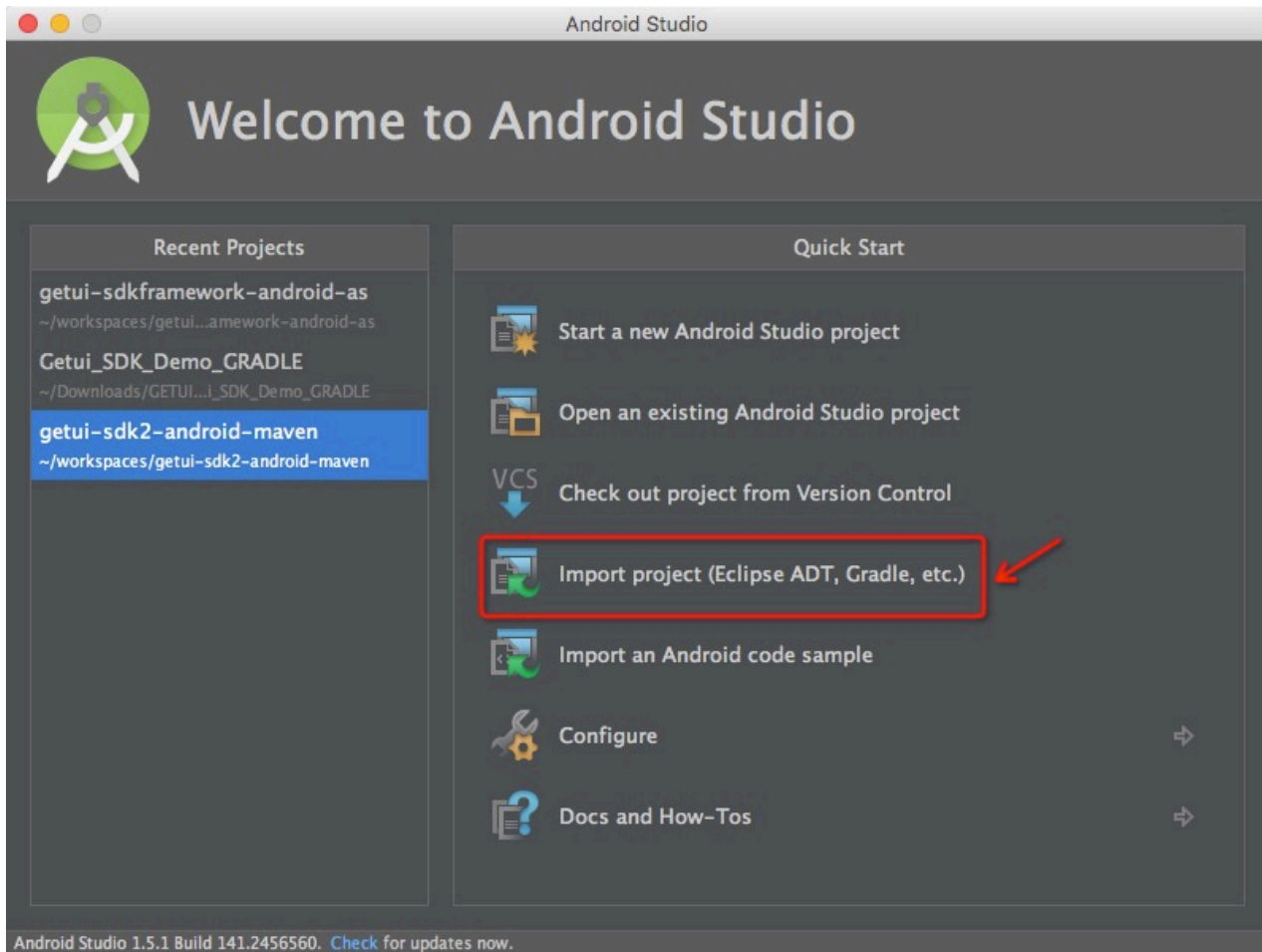
com.ex

2. 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.)`：



3. 添加个推SDK及相关配置

老版本升级到 2.9.5.0 及以上版本注意事项:

1. 替换旧的 `GetuiSDKxxx.jar` , 并删除 `GetuiExt-xxx.jar` 和所有jni相关目录下的 `libgetuiext.so`

2. 删除 `AndroidManifest.xml` 中以下组件相关的配置, 最新的SDK已经不再需要这些组件:

```
com.igexin.sdk.PushServiceUser  
com.igexin.sdk.PushManagerReceiver  
com.igexin.getuiext.activity.GetuiExtActivity  
com.igexin.getuiext.service.PayloadReceiver  
com.igexin.getuiext.service.GetuiExtService
```

3. 删除 `app/src/main/res/layout` 目录下原来旧的布局文件, 包括 `getui_notification.xml` 、 `notification_inc.xml` 和 `increment_popup_dialog.xml` , 请使用最新SDK所提供的 `getui_notification.xml` 即可

4. 请参考本文档重新进行配置集成

3.1 个推Android SDK资料包结构

```

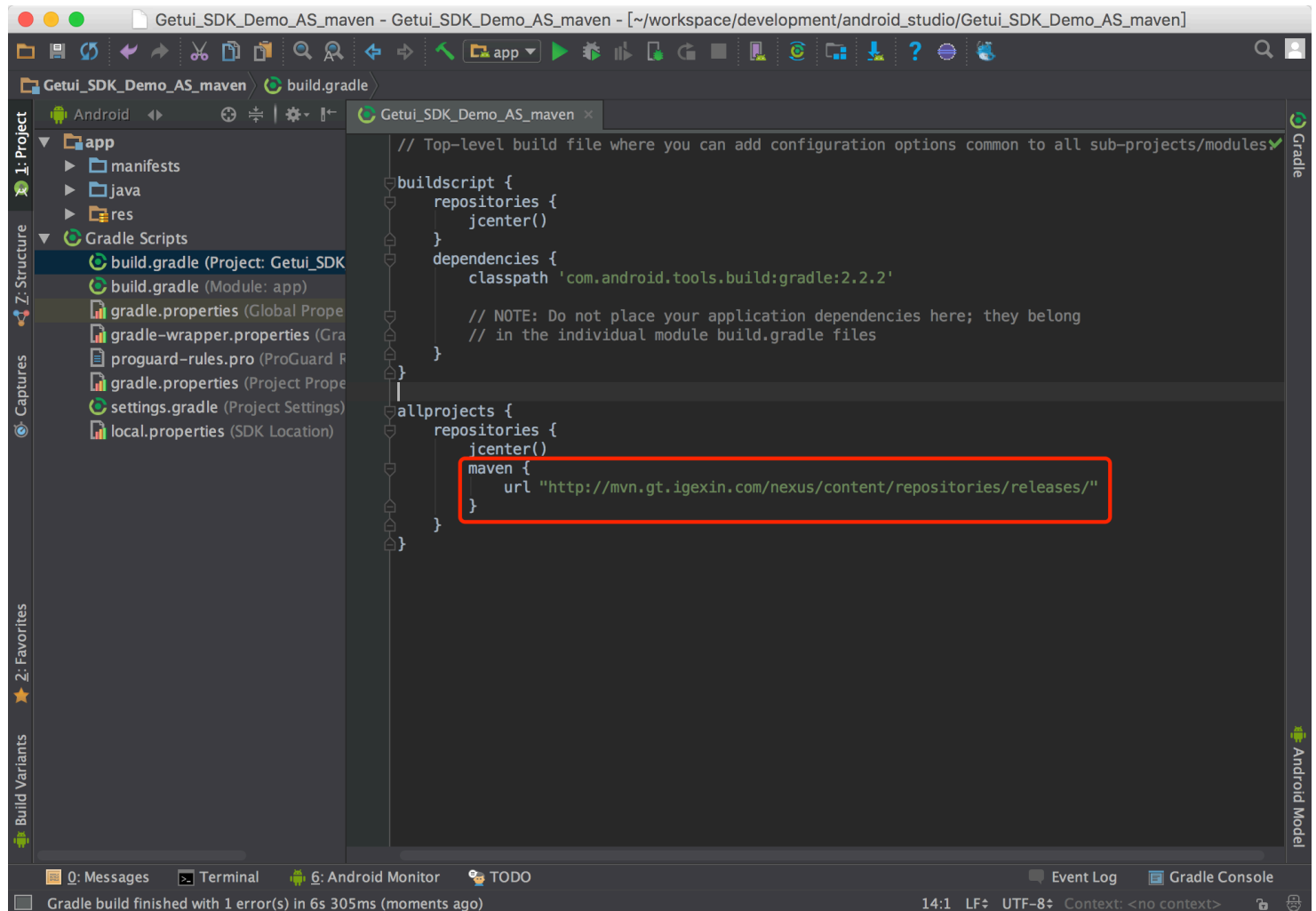
GETUI_ANDROID_SDK/
|- readme.txt (SDK资料包说明)
|- 接入文档/ (Android SDK相关集成文档PDF版本)
|- 资源文件/
|   |- res/
|       |- layout/
|           |- getui_notification.xml (个推SDK所需的布局文件)
|       |- raw
|           |- keep.xml (用于资源保留的描述文件)
|   |- so/ (各 CPU 架构so库)
|       |- arm64-v8a/
|       |- armeabi/
|       |- armeabi-v7a/
|       |- mips/
|       |- mips64/
|       |- x86/
|       |- x86_64/
|   |- GetuiSDK2.10.2.0.jar
|   |- android-support-v4.jar
|- Demo工程/
|   |- Getui_SDK_Demo_AS_maven/ (AndroidStudio快速集成Demo工程)
|   |- Getui_SDK_Demo_AS_official/ (AndroidStudio标准集成Demo工程)
|   |- Getui_SDK_Demo_Eclipse_official/ (Eclipse集成Demo工程)

```

3.2 添加Maven库地址

尽管我们会将最新的个推SDK同步部署在JCenter上，但是为了保障稳定使用，我们建议开发者额外配置个推提供的maven库从而实现更快速的访问。

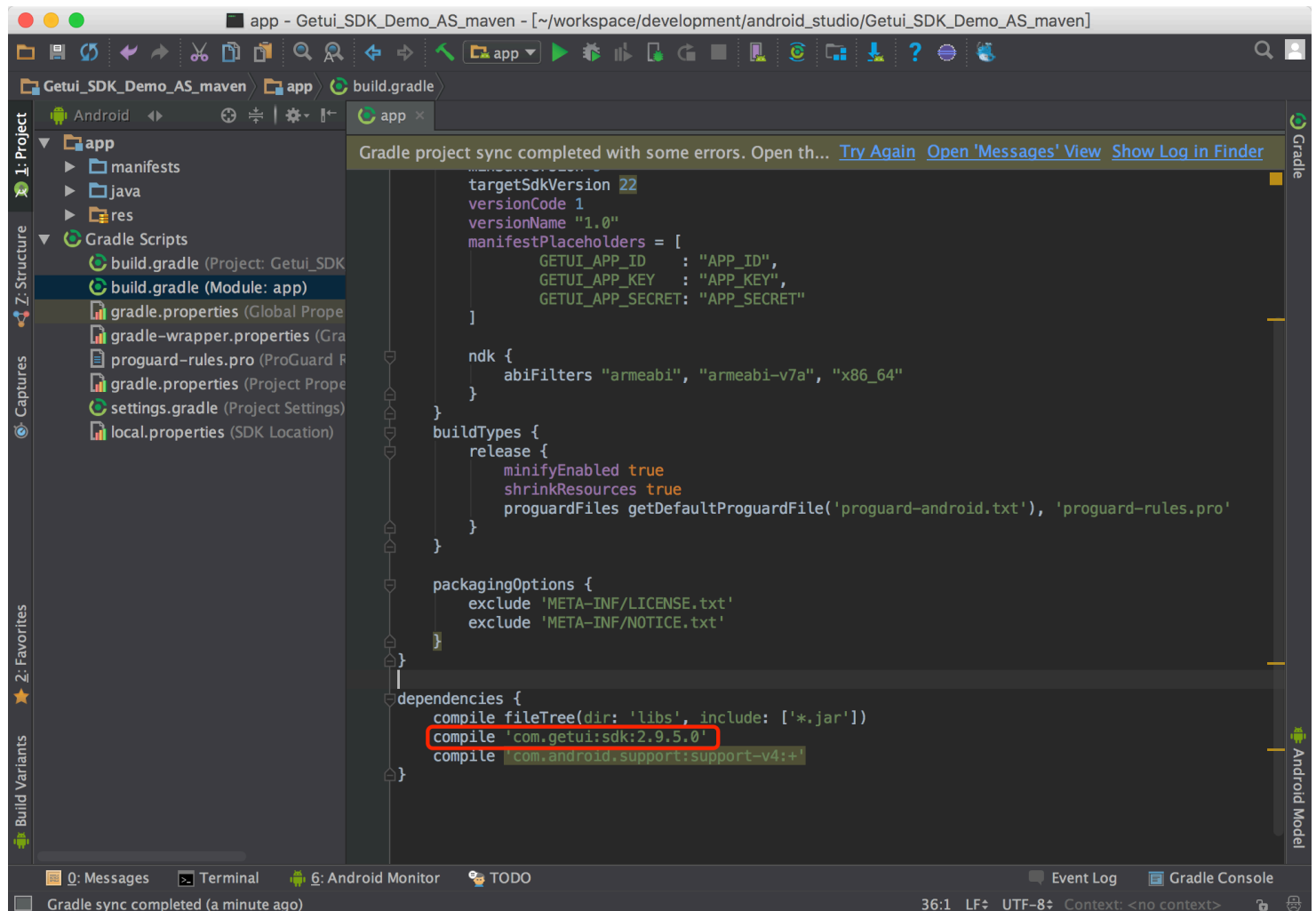
- 在以项目名为命名的顶层 `build.gradle` 文件中，添加个推maven库地址，如下所示：



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

3.3 配置依赖

- 在 `app/build.gradle` 文件中引用个推SDK依赖库，如下图所示：



```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.getui.sdk:2.10.2.0'
    compile 'com.android.support:support-v4:+'
}
```

开发者可以指定`com.getui.sdk`的完整版本号，也可以指定模糊版本号，由`gradle`自动查找符合条件最新的SDK版本，例如：

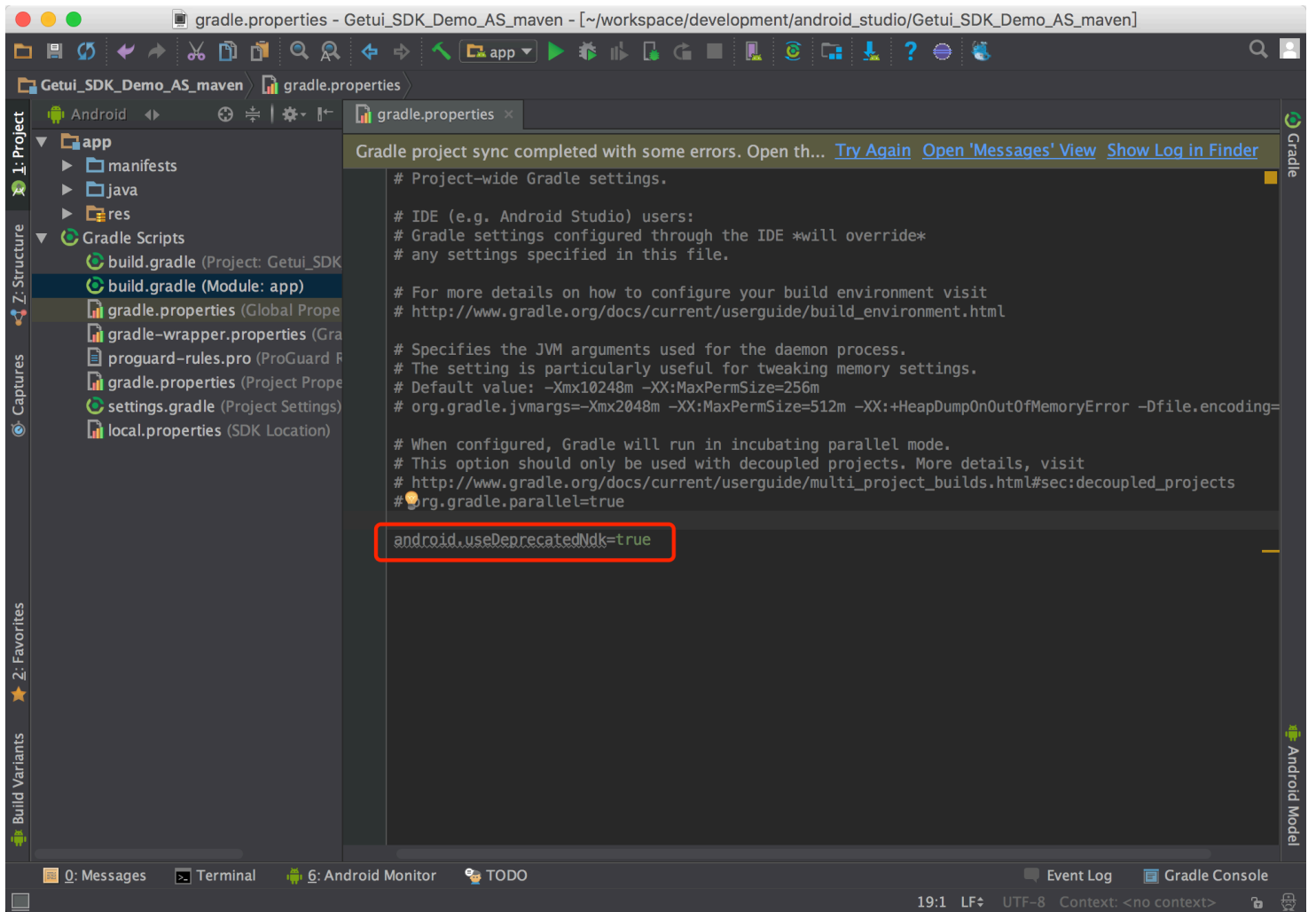
```
compile 'com.getui.sdk:2.+'
```

3.4 配置 so 库

目前个推SDK支持 `armeabi`、`armeabi-v7a`、`arm64-v8a`、`mips`、`mips64`、`x86`、`x86_64` 这几种 CPU 架构，请根据项目情况指定所需的架构。

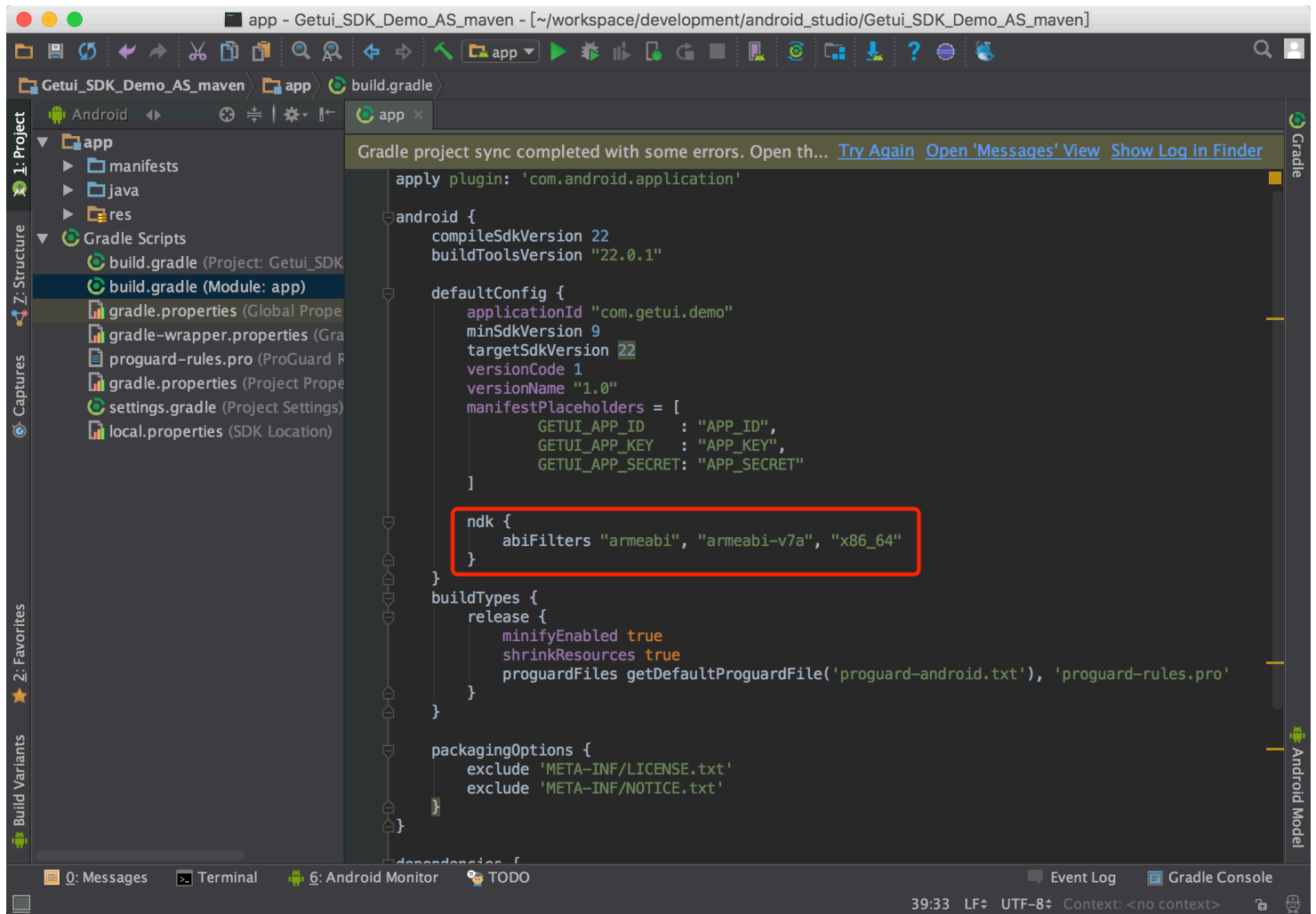
如果项目中包含的其他 so 库只支持其中某几种 cpu 架构，那么应该根据其他 so 库所支持的 CPU 架构的最小集来配置。否则如果在特定架构上未能支持所有 so 库，则很可能导致程序运行异常。切记！

- 在项目根目录下的 `gradle.properties` 文件中配置 `useDeprecatedNdk` 参数，如下图所示：



android.useDeprecatedNdk=true

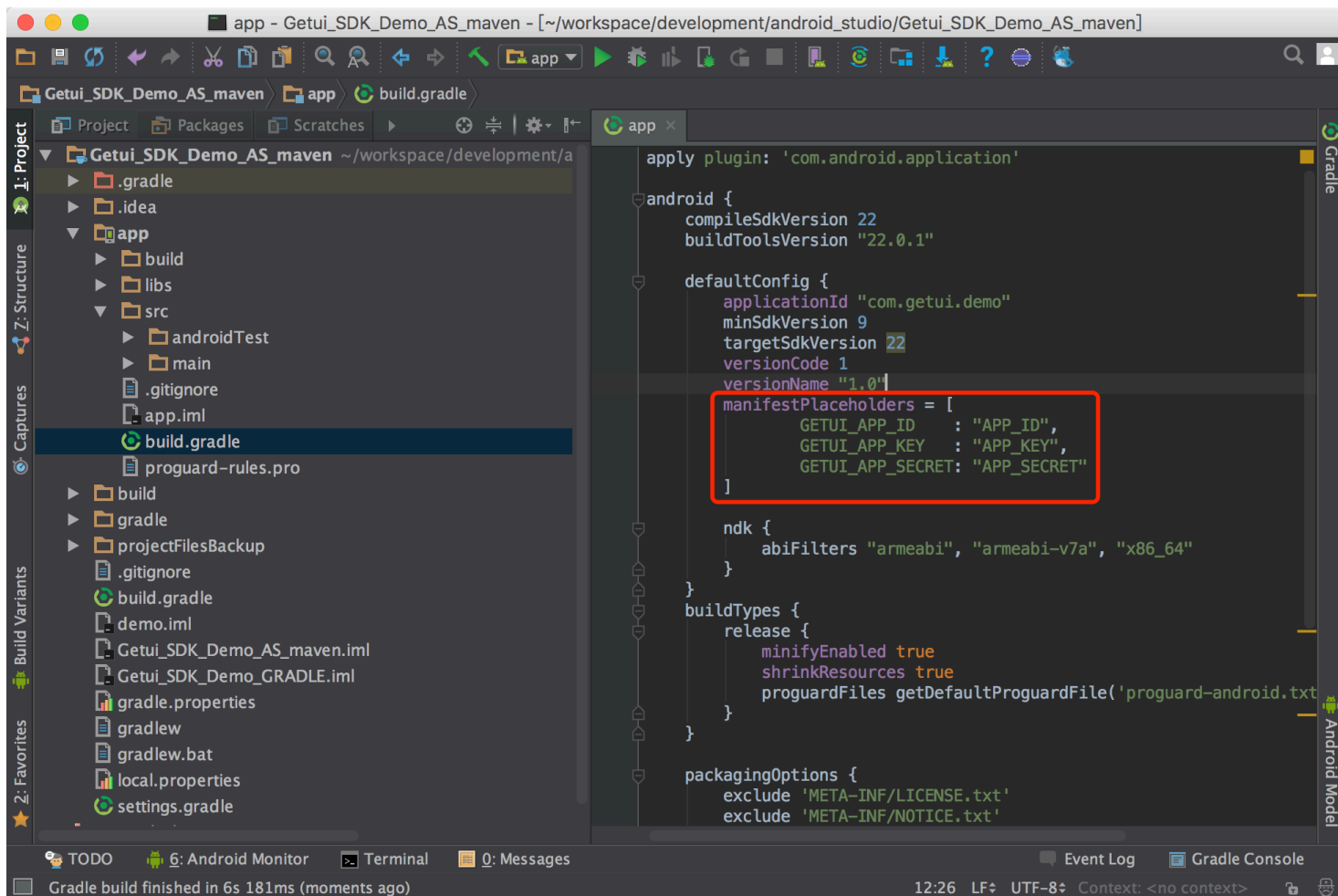
- 在 `app/build.gradle` 文件中的 `android.defaultConfig` 下指定所需的 CPU 架构，如下图所示：



```
android {
    ...
    defaultConfig {
        ...
        ndk {
            abiFilters "armeabi", "armeabi-v7a", "x86_64"
        }
    }
}
```

3.5 配置个推应用参数

- 在 `app/build.gradle` 文件中的 `android.defaultConfig` 下添加 `manifestPlaceholders`，配置个推相关的应用参数（参见【步骤1】），如下图所示：



```
manifestPlaceholders = [
    GETUI_APP_ID : "APP_ID",
    GETUI_APP_KEY : "APP_KEY",
    GETUI_APP_SECRET : "APP_SECRET"
]
```

- 请根据【步骤1】获取到的应用参数进行相应替换 `APP_ID`、`APP_KEY`、`APP_SECRET` 的值

3.6 配置自定义推送服务

为了让推送服务在部分主流机型上更稳定运行，从2.9.5.0版本开始，个推支持第三方应用配置使用自定义Service来作为推送服务运行的载体。

- 在项目源码中添加一个继承自 `Android.app.Service` 的类，参考下列代码实现Service各个生命周期回调方法：

```

package com.getui.demo;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

import com.igexin.sdk.GTServiceManager;

public class DemoPushService extends Service {

    @Override
    public void onCreate() {
        super.onCreate();
        GTServiceManager.getInstance().onCreate(this);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        super.onStartCommand(intent, flags, startId);
        return GTServiceManager.getInstance().onStartCommand(this, intent, flags, startId);
    }

    @Override
    public IBinder onBind(Intent intent) {
        return GTServiceManager.getInstance().onBind(intent);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        GTServiceManager.getInstance().onDestroy();
    }

    @Override
    public void onLowMemory() {
        super.onLowMemory();
        GTServiceManager.getInstance().onLowMemory();
    }
}

```

- 在AndroidManifest.xml中添加上述自定义Service：

```

<service
    android:name="com.getui.demo.DemoPushService"
    android:exported="true"
    android:label="PushService"
    android:process=":pushservice">
</service>

```

3.7 配置可选权限

- 上述接入方式已包含个推服务所需的所有必备权限。除此之外，您还可以配置以下可选权限，以便使用个推3.0提供的电子围栏功能。请在 `AndroidManifest.xml` 的 `<manifest>` 根标签下添加如下配置：

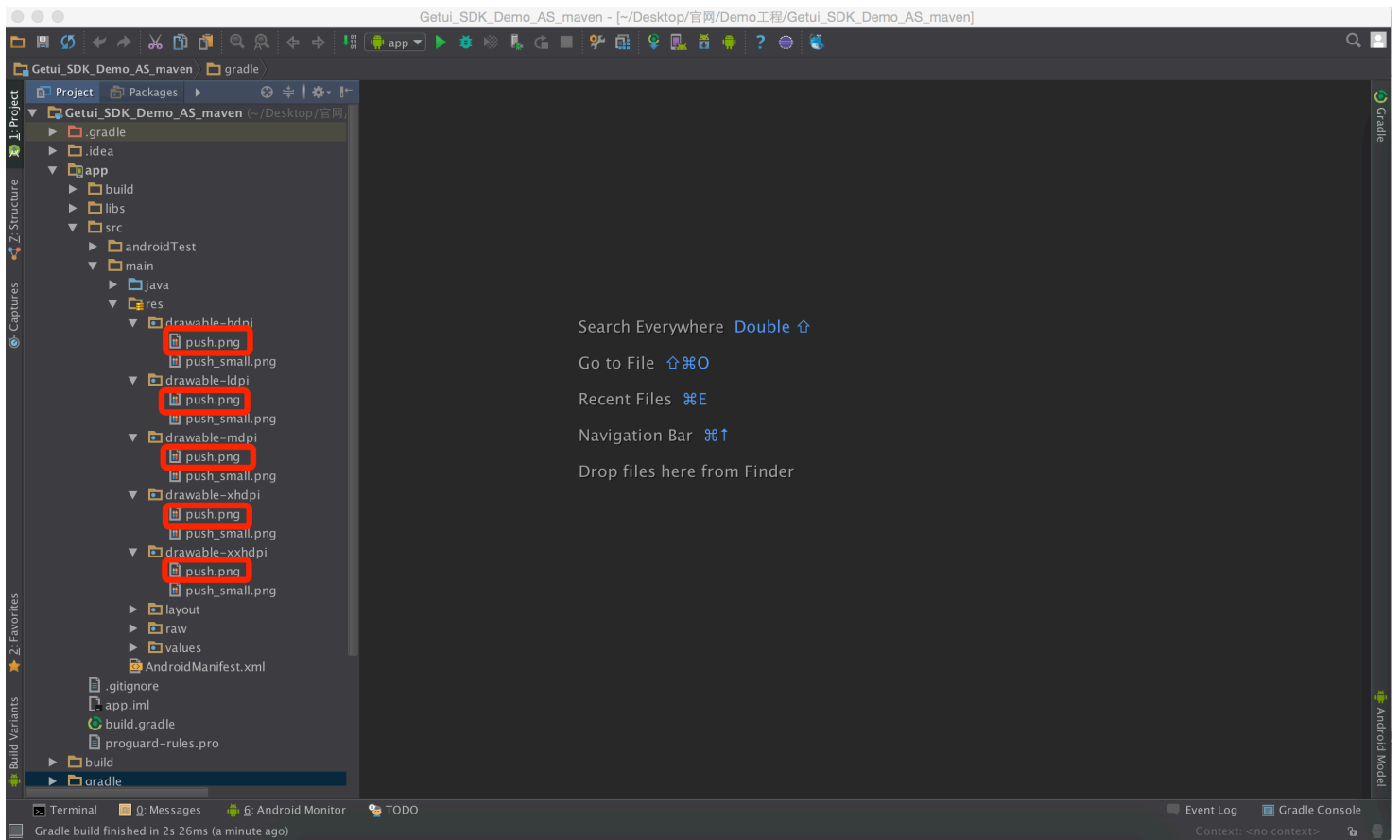
```

<!-- 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" />

```

3.8 导入通知栏图标

- 为了修改默认的通知栏顶部提示小图标，请在资源目录的 `res/drawable-ldpi/`、`res/drawable-mdpi/`、`res/drawable-hdpi/`、`res/drawable-xhdpi/`、`res/drawable-xxhdpi/` 等各分辨率目录下，放置相应尺寸的文件名为 **push.png** 图片，如图所示：



- 建议的 `push.png` 图片尺寸如下：

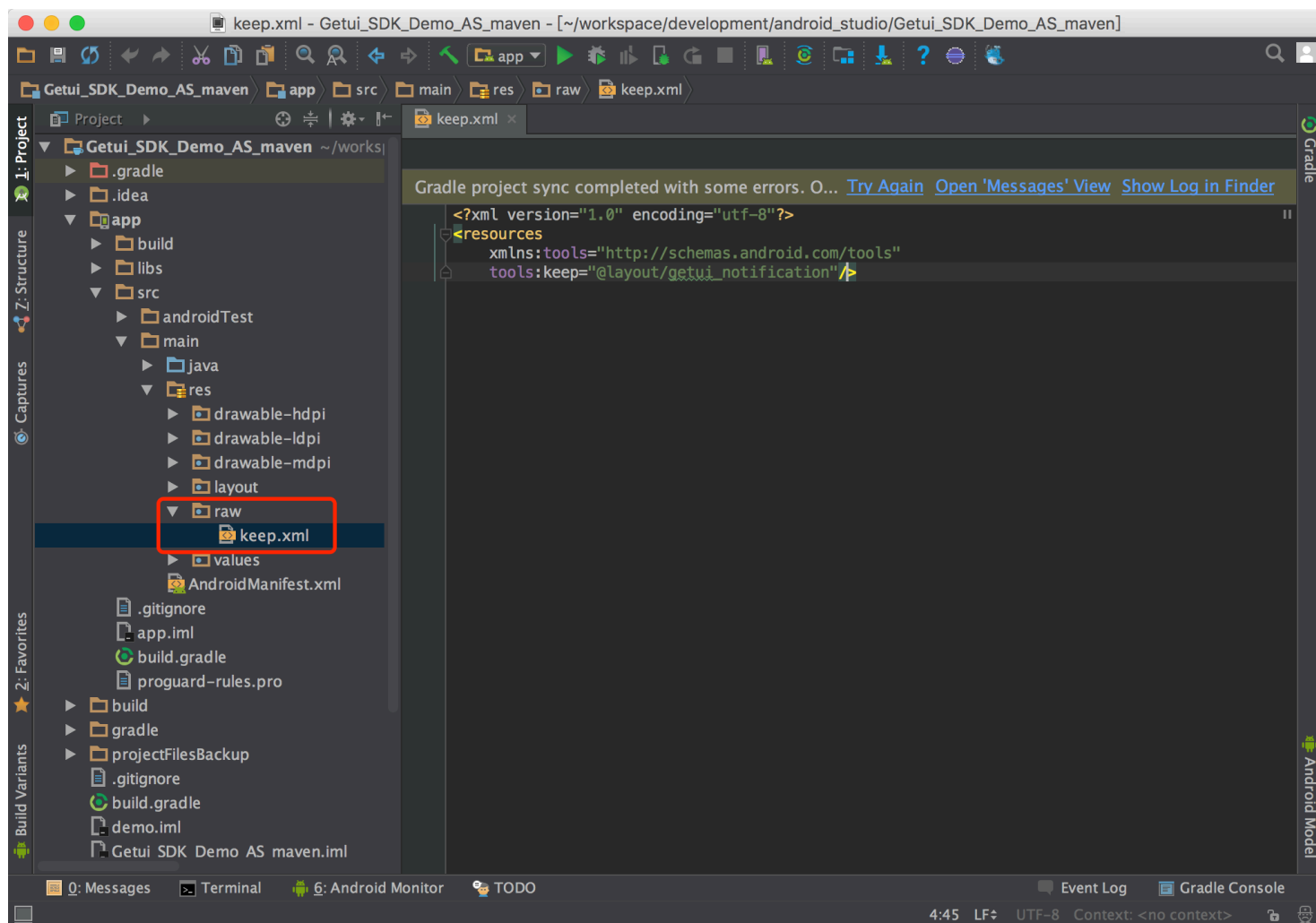
ldpi:	48*48
mdpi:	64*64
hdpi:	96*96
xhdpi:	128*128
xxhdpi:	192*192

- 该图标将会作为通知图标展示在通知栏顶部，如下所示：



3.9 资源精简配置

- 如果您的工程启用了资源精简，即如果在 `app/build.gradle` 的 `android.buildTypes.release` 下配置了 `shrinkResources true`，为了避免个推SDK所需资源被错误精简导致功能异常，需要在项目资源目录 `res/raw` 中添加 `keep.xml` 文件，路径如下：



- 在SDK资料包中 `GETUI_ANDROID_SDK/资源文件/raw` 目录下有 `keep.xml` 示例文件，完整内容如下：

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

- 如果你的项目工程已经使用了 `keep.xml`，则只需在 `tools:keep` 中增加对 `@layout/getui_notification` 的声明，例如：

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

4. 编写集成代码

4.1 初始化SDK

我们建议应用开发者在Activity或服务类中调用个推SDK的初始化方法，确保SDK在各种情况下都能正常运行。一般情况下可以在主Activity的 `onCreate()` 或者 `onResume()` 方法中调用，也可以在多个主要界面Activity的 `onCreate()` 或 `onResume()` 方法中调用。反复调用SDK初始化并不会有什么副作用。

- 在应用的 Activity 里导入 `PushManager` 类，如下所示：

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

- 然后在 Activity 的 `onCreate()` 或者 `onResume()` 方法中调用个推SDK初始化方法。如果使用了自定义推送服务，初始化方法还需要传入新的自定义

推送服务名：

```
// com.getui.demo.DemoPushService 为第三方自定义推送服务
PushManager.getInstance().initialize(this.getApplicationContext(), com.getui.demo.DemoPushService.class);
```

4.2 接收推送服务事件

从2.9.5.0版本开始，为了解决小概率发生的Android广播丢失问题，我们推荐应用开发者使用新的IntentService方式来接收推送服务事件（包括CID获取通知、透传消息通知等）

- 在项目源码中添加一个继承自com.igexin.sdk.GTIntentService的类，用于接收CID、透传消息以及其他推送服务事件。请参考下列代码实现各个事件回调方法：

```
package com.getui.demo;

import android.content.Context;
import android.os.Message;
import android.util.Log;

import com.igexin.sdk.GTIntentService;
import com.igexin.sdk.PushConsts;
import com.igexin.sdk.PushManager;
import com.igexin.sdk.message.FeedbackCmdMessage;
import com.igexin.sdk.message.GTCmdMessage;
import com.igexin.sdk.message.GTTransmitMessage;
import com.igexin.sdk.message.SetTagCmdMessage;

/**
 * 继承 GTIntentService 接收来自个推的消息，所有消息在线程中回调，如果注册了该服务，则务必要在 AndroidManifest中声明，否则无法接受消息<br>
 * onReceiveMessageData 处理透传消息<br>
 * onReceiveClientId 接收 cid <br>
 * onReceiveOnlineState cid 离线上线通知 <br>
 * onReceiveCommandResult 各种事件处理回执 <br>
 */
public class DemoIntentService extends GTIntentService {

    public DemoIntentService() {

    }

    @Override
    public void onReceiveServicePid(Context context, int pid) {

    }

    @Override
    public void onReceiveMessageData(Context context, GTTransmitMessage msg) {

    }

    @Override
    public void onReceiveClientId(Context context, String clientId) {
        Log.e(TAG, "onReceiveClientId -> " + "clientId = " + clientId);
    }

    @Override
    public void onReceiveOnlineState(Context context, boolean online) {

    }

    @Override
    public void onReceiveCommandResult(Context context, GTCmdMessage cmdMessage) {

    }

}
```

- 在 `AndroidManifest.xml` 中配置上述 `IntentService` 类：

```
<service android:name="com.getui.demo.DemoIntentService" />
```

- 在个推SDK初始化后，注册上述 `IntentService` 类：

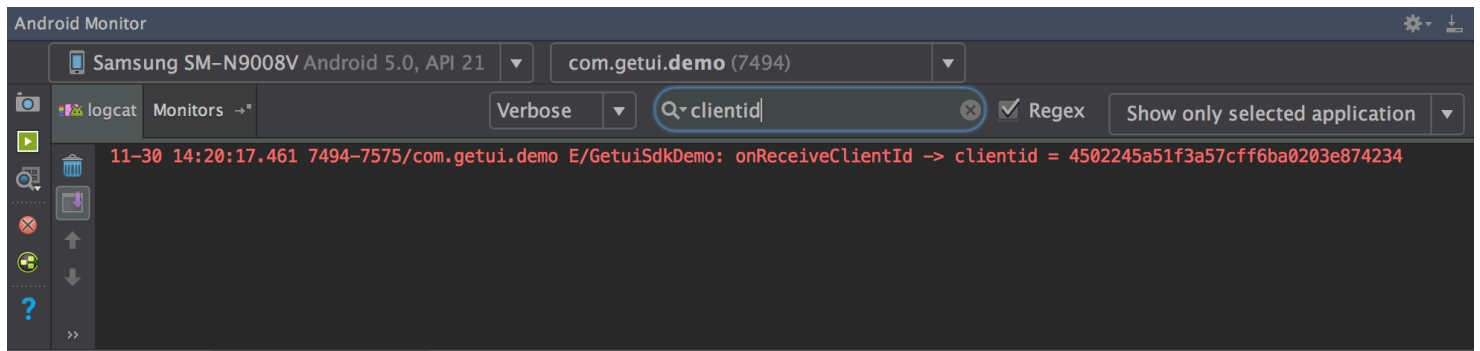
```
// com.getui.demo.DemoIntentService 为第三方自定义的推送服务事件接收类
PushManager.getInstance().registerPushIntentService(this.getApplicationContext(), com.getui.demo.DemoIntentService.class);
```

关于原有广播方式和新的`IntentService`方式兼容性说明：


1. 如果调用了`registerPushIntentService`方法注册自定义`IntentService`，则SDK仅通过`IntentService`回调推送服务事件；
2. 如果未调用`registerPushIntentService`方法进行注册，则原有的广播接收器仍然可以继续使用。

5. 测试

- 连接手机或启动Android模拟器，编译运行你的工程，查看logcat信息。在搜索框中输入 `clientid`，如果能显示 `clientid is xxx` 日志，则说明个推SDK已经成功运行起来了：



- 登录 [个推开发者平台](#)，点击【创建推送】，进入待测试应用的推送通知界面：

个推 App De...

创建推送

推送通知

透传消息

分组对比测试

数据统计

推送记录

推送数据

推送统计

用户数据

配置管理

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： 0/20

* 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： 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



- 依次填写 通知标题 和 通知内容 ，点击 发送 按钮即可向该推送应用名下所有CID推送通知消息。具体推送操作方法详见：[创建推送通知](#)
- 如果手机或模拟器收到消息，显示下图所示通知，那么恭喜您，个推SDK接入测试已经成功完成！

15:58

3月14日星期一



欢迎使用个推!

个推免费啦~~~



Google



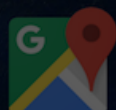
Play 商店



个推演示



Android Pay



地图

