

# 发布Android版APP

在典型的开发周期中，您将使用`flutter run`命令行或者IntelliJ中通过工具栏运行和调试按钮进行测试。默认情况下，Flutter构建应用程序的`debug`版本。

当您准备好为Android准备的`release`版时，例如要发布到应用商店，请按照此页面上的步骤操作。

- [检查 App Manifest](#)
- [查看构建配置](#)
- [添加启动图标](#)
- [app签名](#)
  - [创建 keystore](#)
  - [引用应用程序中的keystore](#)
  - [在gradle中配置签名](#)
- [构建一个发布版 \( release \) APK](#)
- [在设备上安装发行版APK](#)
- [将APK发布到Google Play商店](#)

## 检查 App Manifest

查看默认[应用程序清单](#)文件(位于`<app dir>/android/app/src/main/`中的`AndroidManifest.xml`文件)，并验证这些值是否正确，特别是：

- `application`: 编辑 `application` 标签，这是应用的名称。
- `uses-permission`: 如果您的应用程序代码不需要Internet访问，请删除`android.permission.INTERNET`权限。标准模板包含此标记是为了启用Flutter工具和正在运行的应用程序之间的通信。

## 查看构建配置

Review the default [Gradle build file][gradlebuild] file `build.gradle` located in `<app dir>/android/app/` and verify the values are correct, especially: 查看默认[Gradle 构建文件][gradlebuild]”`build.gradle`”，它位于`<app dir>/android/app/`，验证这些值是否正确，尤其是：

- `defaultConfig`:
  - `applicationId`: 指定始终唯一的 (Application Id)[appid](#)
  - `versionCode` & `versionName`: 指定应用程序版本号和版本号字符串。有关详细信息，请参考[版本文档](#)
  - `minSdkVersion` & `targetSdkVersion`: 指定最低的API级别以及应用程序设计运行的API级别。有关详细信息，请参阅[版本文档](#)中的API级别部分。

## 添加启动图标

当一个新的Flutter应用程序被创建时，它有一个默认的启动器图标。要自定义此图标：

1. 查看[Android启动图标](#) 设计指南，然后创建图标。
2. 在`<app dir>/android/app/src/main/res/`目录中，将图标文件放入使用[配置限定符](#)命名的文件夹中。默认`mipmap`文件夹演示正确的命名约定。
3. 在`AndroidManifest.xml`中，将[application](#)标记的`android:icon`属性更新为引用上一步中的图标（例如`<application android:icon="@mipmap/ic_launcher" ...`）。
4. 要验证图标是否已被替换，请运行您的应用程序并检查应用图标

## app签名

### 创建 keystore

如果您有现有keystore，请跳至下一步。如果没有，请通过在运行以下命令来创建一个：  
`keytool -genkey -v -keystore ~/key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias key`

**注意：**保持文件私密; 不要将它加入到公共源代码控制中。

**注意：**`keytool`可能不在你的系统路径中。它是Java JDK的一部分，它是作为Android Studio的一部分安装的。有关具体路径，请百度。

### 引用应用程序中的keystore

创建一个名为`<app dir>/android/key.properties`的文件，其中包含对密钥库的引用：

```
storePassword=<password from previous step>
keyPassword=<password from previous step>
keyAlias=key
storeFile=<location of the key store file, e.g. /Users/<user name>/key.jks>
```

**注意：**保持文件私密; 不要将它加入公共源代码控制中

## 在gradle中配置签名

通过编辑<app dir>/android/app/build.gradle文件为您的应用配置签名

### 1. 替换:

```
android {
```

为:

```
def keystorePropertiesFile = rootProject.file("key.properties")
def keystoreProperties = new Properties()
keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
```

```
android {
```

### 2. 替换:

```
buildTypes {
    release {
        // TODO: Add your own signing config for the release build.
        // Signing with the debug keys for now, so `flutter run --release` works.
        signingConfig signingConfigs.debug
    }
}
```

为:

```
signingConfigs {
    release {
        keyAlias keystoreProperties['keyAlias']
        keyPassword keystoreProperties['keyPassword']
        storeFile file(keystoreProperties['storeFile'])
        storePassword keystoreProperties['storePassword']
    }
}
buildTypes {
    release {
        signingConfig signingConfigs.release
    }
}
```

现在，您的应用的release版本将自动进行签名。

## 构建一个发布版（release）APK

本节介绍如何构建发布版（release）APK。如果您完成了前一节中的签名步骤，则会对APK进行签名。

使用命令行:

1. `cd <app dir>` (<app dir> 为您的工程目录).
2. 运行 `flutter build apk` (`flutter build` 默认会包含 `--release` 选项).

打包好的发布APK位于 `<app dir>/build/app/outputs/apk/app-release.apk`。

## 在设备上安装发行版APK

按照以下步骤在已连接的Android设备上安装上一步中构建的APK  
使用命令行:

1. 用USB您的Android设备连接到您的电脑
2. `cd <app dir> .`
3. 运行 `flutter install` .

## 将APK发布到Google Play商店

将应用的release版发布到Google Play商店的详细说明，请参阅 [Google Play publishing documentation](#). (国内不存在的，但你可以发布到国内的各种应用商店)