Build and release an android app

This article provides a step-by-step guide for building an APK or bundle for your app and releasing it on play store.

In this guide you will need to run several commands from the terminal. Make sure to `cd` into your app directory and run all commands from there. Easiest way would be to press terminal, however you are free to use the terminal of your choice for your OS.

Signing the app

To publish on the Play Store, you need to give your app a digital signature. Use the following instructions to sign your app.

Create a keystore

If you have an existing keystore, skip to the next step. If not, create one by running the following at the command line:

- On Mac/Linux, use the following command:

```
keytool -ger
```

command:

```
keytool -ger
```

This command stores the **key.jks** file in your home directory. If you want to store it elsewhere, change the argument you pass to the -keystore parameter. Make sure to save this file as it will be required when submitting updates in the future.

Entering keyfile credentials

Open `android/key.properties` file and enter password and location to keystore file from previous steps.

Building the app for release

You have two options when building for release on android:

- App bundle (preferred)
- APK

Note: The Google Play Store prefers the app bundle format. For more information, see Android App Bundle and About Android App Bundles.

Build an app bundle

This section describes how to build a release app bundle will be signed. At this point, you might consider obfuscating your Dart code to make it more difficult to reverse engineer. Obfuscating your code involves adding a couple flags to your build command, and maintaining additional files to de-obfuscate stack traces.

From the command line run:

```
flutter build app
```

The release bundle for your app is created at `/build/app/outputs/bundle/release/app.aab`.

By default, the app bundle contains code compiled for armeabi-v7a (ARM 32-bit), arm64-v8a (ARM 64-bit), and x86-64 (x86 64-bit).

An app bundle can be tested in multiple ways—this section describes two.

Offline using the bundle tool

1. If you haven't done so already, download `bundletool` from the GitHub repository.
2. Generate a set of APKs from your app bundle.
3. Deploy the APKs to connected devices.

Online using google play

1. Upload your bundle to Google Play to test it. You can use the internal test track, or the alpha or beta channels to test the bundle before

2. Follow these steps to upload your bundle to the Play Store.

Build an APK

Although app bundles are preferred over APKs, there are stores that don't yet support app bundles. In this case, build a release APK for each target ABI (Application Binary Interface).

If you completed the signing steps, the APK will be signed. At this point, you might consider obfuscating your Dart code to make it more difficult to reverse engineer. Obfuscating your code involves adding a couple flags to your build command.

From the command line run:

```
flutter build apk
```

results in three APK files:

- `/build/app/outputs/apk/release/app-armeabi-v7a-release.apk`
- `/build/app/outputs/apk/release/app-arm64-v8a-release.apk`
- `/build/app/outputs/apk/release/app-x86_64-release.apk`

Removing the `--split-per-abi` flag results in a fat APK that contains your code compiled for *all* the target ABIs. Such APKs are larger in size than their split counterparts, causing the user to download native binaries that are not applicable to their device's architecture.

Install an APK on a device

Follow these steps to install the APK on a

From the command line:

1. Connect your Android device to your computer with a USB cable.
2. Run `flutter install`.

If you can't find something in this guide you can check the official guide from flutter, note that a number of steps from flutter guide can be skipped as they are already done in BeDrive flutter.

Have more questions? Submit a Request

Was this ✓ YES