Flutter Deployment & Publishing: Google Play & App Store

A practical guide for developers to build, sign, and publish Flutter apps to mobile stores. Learn how to deploy production-ready apps safely and efficiently.

Website reference: haas.dev | https://dev-roast-app.vercel.app

---

Introduction

Publishing a Flutter app is the final step in turning your code into a real, usable mobile app. Many beginners struggle with deployment because of signing, store requirements, and platform differences. This guide explains everything step by step for both **Google Play Store (Android)** and **Apple App Store (iOS)**.

By following this guide, you'll learn how to prepare your app for release, meet store requirements, and avoid common errors.

Step 1: Preparing Your Flutter App for Release

Before publishing, ensure your app is **production-ready**:

1. **Versioning**
    - Update `version` and `build number` in `pubspec.yaml`:

```
version: 1.0.0+1
```

        - `1.0.0` → version visible to users
        - `+1` → internal build number
2. **App Icon & Splash Screen**
    - Use the `flutter_launcher_icons` package to generate icons for all sizes.
    - Configure `launch_background.xml` (Android) and `LaunchScreen.storyboard` (iOS) for splash screens.
3. **App Name & Bundle Identifier**
    - Android: Update `applicationId` in `android/app/build.gradle`.
    - iOS: Update `Bundle Identifier` in Xcode (`Runner` target → `General`).

4. **Remove Debug Code**
    - Ensure no `print()` statements remain.
    - Disable debugging flags:

```
debugShowCheckedModeBanner: false
```

Step 2: Android Deployment (Google Play)

2.1 Generate a Release APK or App Bundle

- Navigate to your project folder in terminal:

```
flutter build appbundle --release
```

- Output: `build/app/outputs/bundle/release/app-release.aab`

  Note: Google Play recommends `.aab` instead of `.apk` for optimized delivery.

2.2 Sign the App

1. Generate a keystore:

```
keytool -genkey -v -keystore ~/my-release-key.jks -keyalg RSA -keysize 2048
```

2. Store keystore safely; add path to `android/key.properties`:

```
storePassword=<password>
keyPassword=<password>
keyAlias=my-key
storeFile=/path/to/my-release-key.jks
```

3. Configure `build.gradle` to use this keystore for release builds.

2.3 Test Release Build

```
flutter install --release
```

Test thoroughly on a real device.

2.4 Upload to Google Play

1. Sign up for Google Play Console.
2. Create a new app → Fill required details (title, description, screenshots, privacy policy).
3. Upload `.aab` file → Complete content rating, pricing, distribution.
4. Submit for review → Wait for approval (usually a few hours to a day).

Step 3: iOS Deployment (Apple App Store)

3.1 Set Up Certificates & Provisioning

- Open project in **Xcode**.
- Go to `Runner` → `Signing & Capabilities`.
- Use your **Apple Developer Account** to automatically manage signing.
- Ensure proper **Bundle Identifier** and **team selection**.

3.2 Build the App

- In terminal:

```
flutter build ios --release
```

- Open `ios/Runner.xcworkspace` in Xcode → Product → Archive.

3.3 Upload to App Store

1. Use **Xcode Organizer** → Upload your archive to **App Store Connect**.
2. Fill app metadata, screenshots, and privacy policy.
3. Submit for review → Approval usually takes 1–3 days.

Step 4: Post-Publish Checklist

1. Test the live app on multiple devices.
2. Monitor analytics & crash reports (Firebase Crashlytics recommended).
3. Update regularly with bug fixes and improvements.
4. Respond to user reviews to maintain credibility.

Step 5: Common Issues & Fixes

Issue	Fix
App rejected due to **missing icons**	Check `flutter_launcher_icons` and add all required sizes
Version conflicts	Increment version and build number in `pubspec.yaml`
Provisioning errors (iOS)	Revoke & regenerate certificates or switch to automatic signing
App crashes on release	Test with `flutter run --release` on real device

Summary / Key Takeaways

- Always increment **version/build numbers** before publishing.
- Use **release builds**, not debug builds.
- Follow store guidelines: screenshots, privacy policy, content rating.
- Test on **real devices** before submitting.
- Deployment requires both technical steps (signing, building) and non-technical steps (metadata, description, compliance).

Visit **haas.dev** for more resources and guides.
https://dev-roast-app.vercel.app