

Flutter Fundamentals Creating Production Builds



Peter Kassenaar –
info@kassenaar.com



Creating a Production Build

Deploying your app to various platforms

Deployment



- Many deployments possible!
 - **Android apps** – publication in *Google Play Store*
 - **iOS apps** – publication in *Apple App Store*
 - **macOS apps** – *Apple App Store*
 - **Linux apps** – publication to *Snap Store* or other channel
 - **Windows apps** – publication on *internal network* (mostly)
 - **Web apps** – publication on *(internal) web server*
- ...



Follow steps for **YOUR** platform

- Read the recipes
- Often:
 - Creating **launcher icons**
 - **Signing your app** with a specific key
 - **Build** for release
 - **Publish** to destination
 - ...

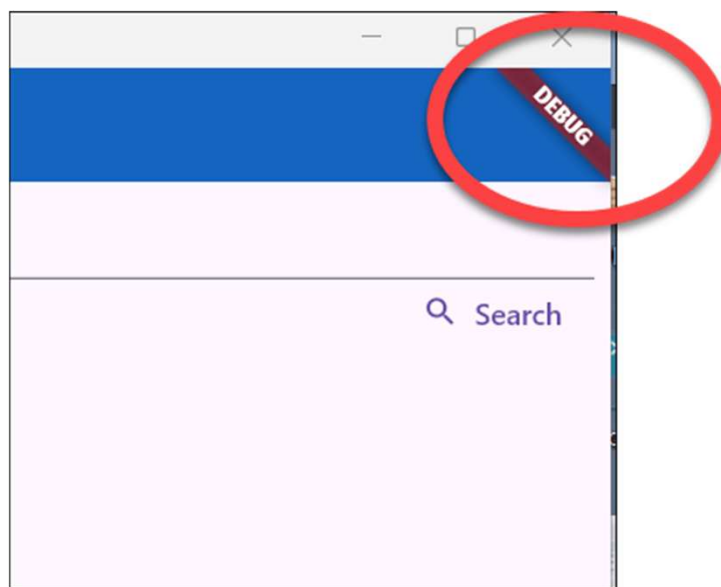
- Following tasks:
- Add a launcher icon
 - Enable Material Components
 - Sign the app
 - Shrink your code with R8
 - Enable multidex support
 - Review the app manifest
 - Review the build configuration
 - Build the app for release
 - Publish to the Google Play Store
 - Update the app's version number
 - Android release FAQ

Example



Building for Windows

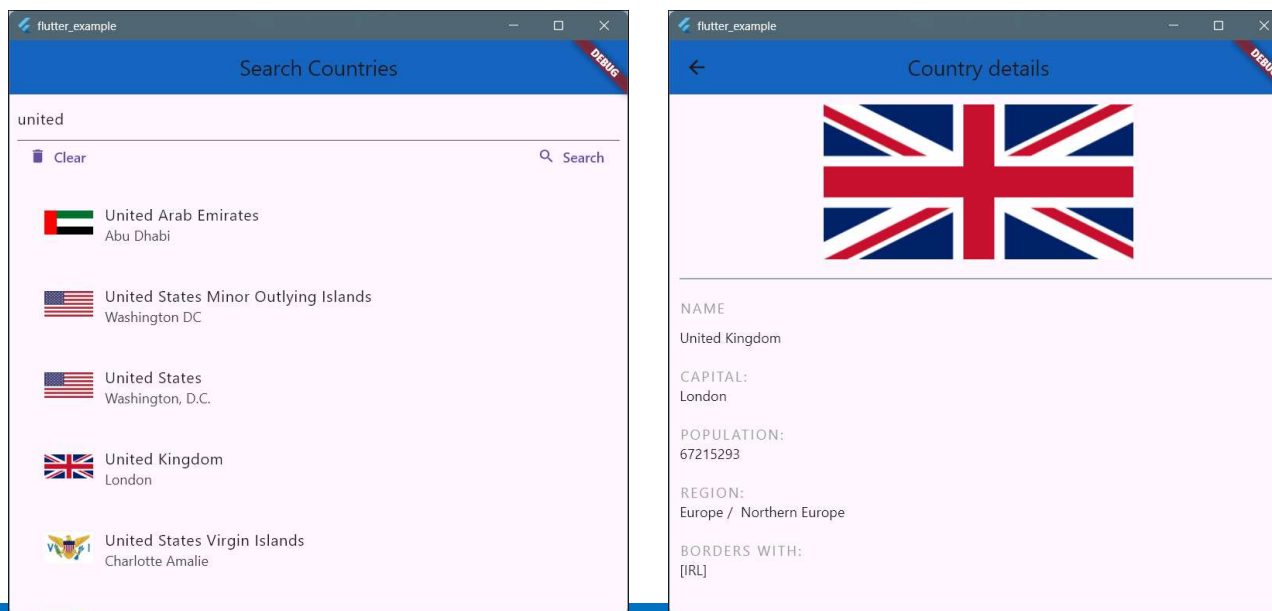
- Example – building a [Windows app](#)
- Default – when running from IDE, [debug version](#)
- Goal – create a [standalone app](#) (not a debug version)





Prerequisites

- We're building a deployment version of `../examples/_310-routing-detail`
 - There are other apps available, this is **just a choice**
 - **NO signing** for publishing in Windows Store





General workflow

Use the `flutter build <platform>` command

Available subcommands:

<code>aar</code>	Build a repository containing an AAR and a POM file.
<code>apk</code>	Build an Android APK file from your app.
<code>appbundle</code>	Build an Android App Bundle file from your app.
<code>bundle</code>	Build the Flutter assets directory from your app.
<code>web</code>	Build a web application bundle.
<code>windows</code>	Build a Windows desktop application.

Run "flutter help" to see global options.

`~\Desktop\flutter_example`

We are building a Windows application



- Command `flutter build windows`

- Use your IDE or a command line terminal

- Executable is stored in

`build\windows\x64\runner\Release\<appName>.exe`

```
~\Desktop\flutter_example
```

```
flutter build windows
```

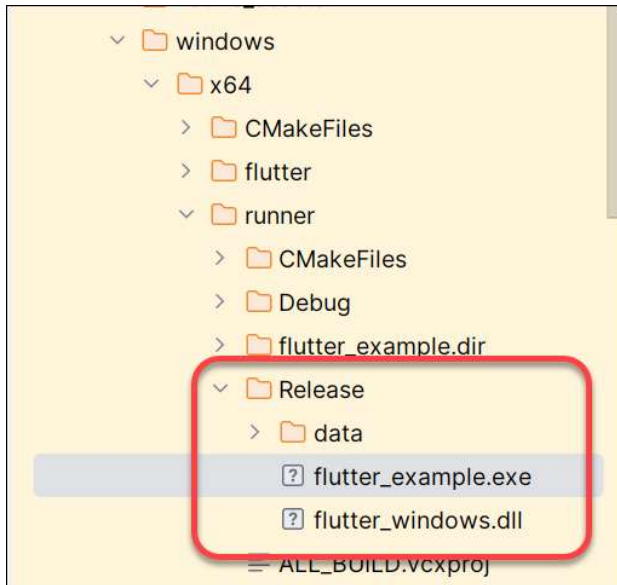
```
Building Windows application...
```

```
35,4s
```

```
✓ Built build\windows\x64\runner\Release\flutter_example.exe
```

```
~\Desktop\flutter_example
```


Deployment of your app

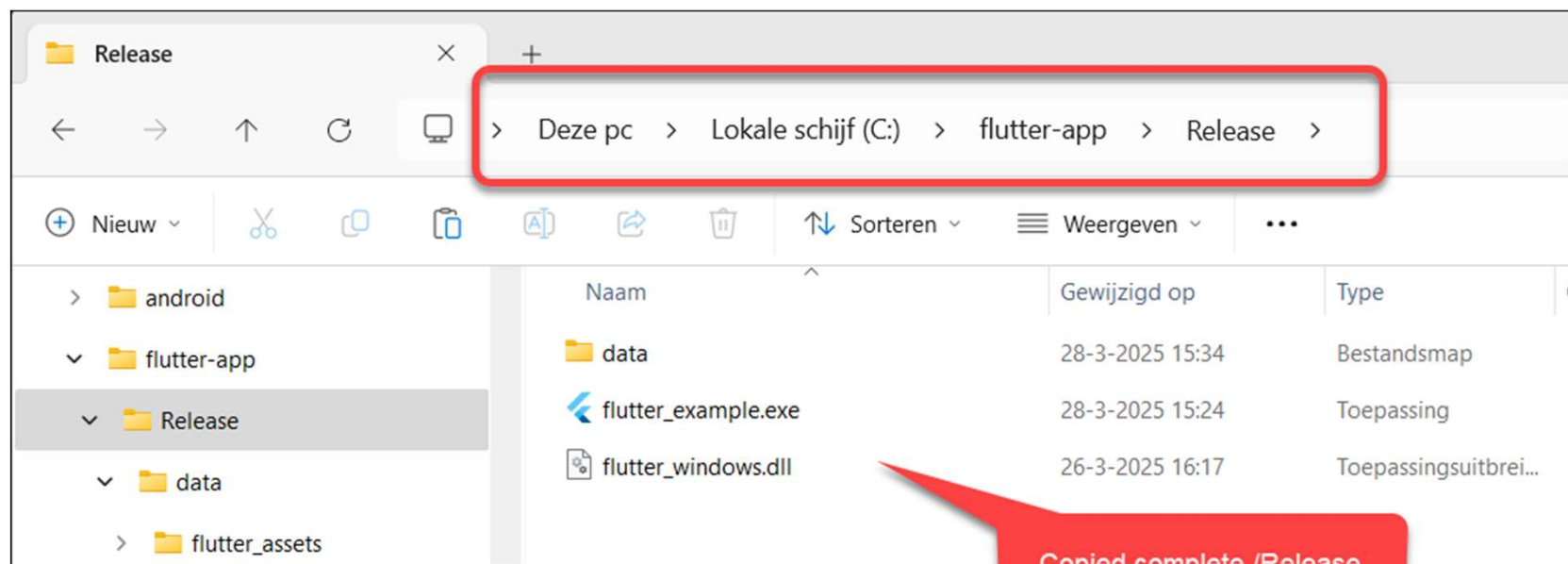


- Distribute the **complete contents** of the generated Release directory.
 - NOT just the generated .exe file
- The .exe **depends on**
 - DLLs like `flutter_windows.dll`, and possibly more
 - data folder with assets, ICU data, AOT runtime, etc.
- Otherwise your app will crash or show runtime errors.



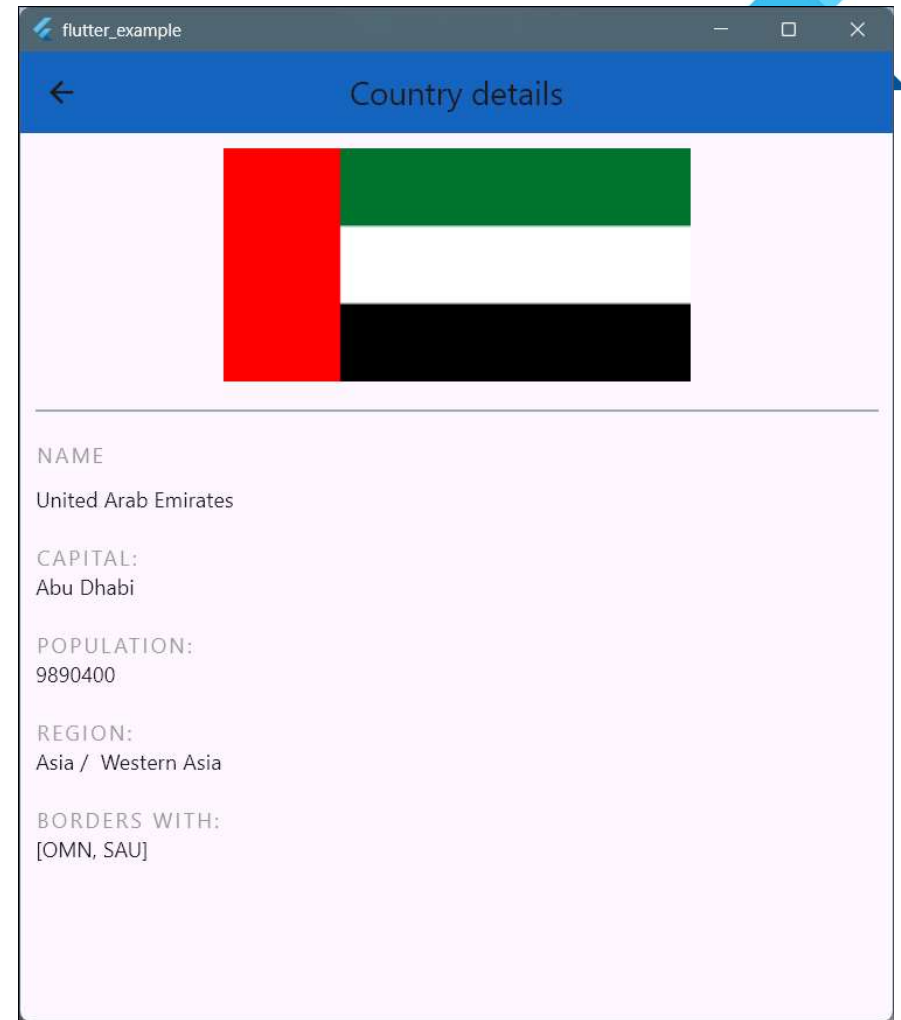
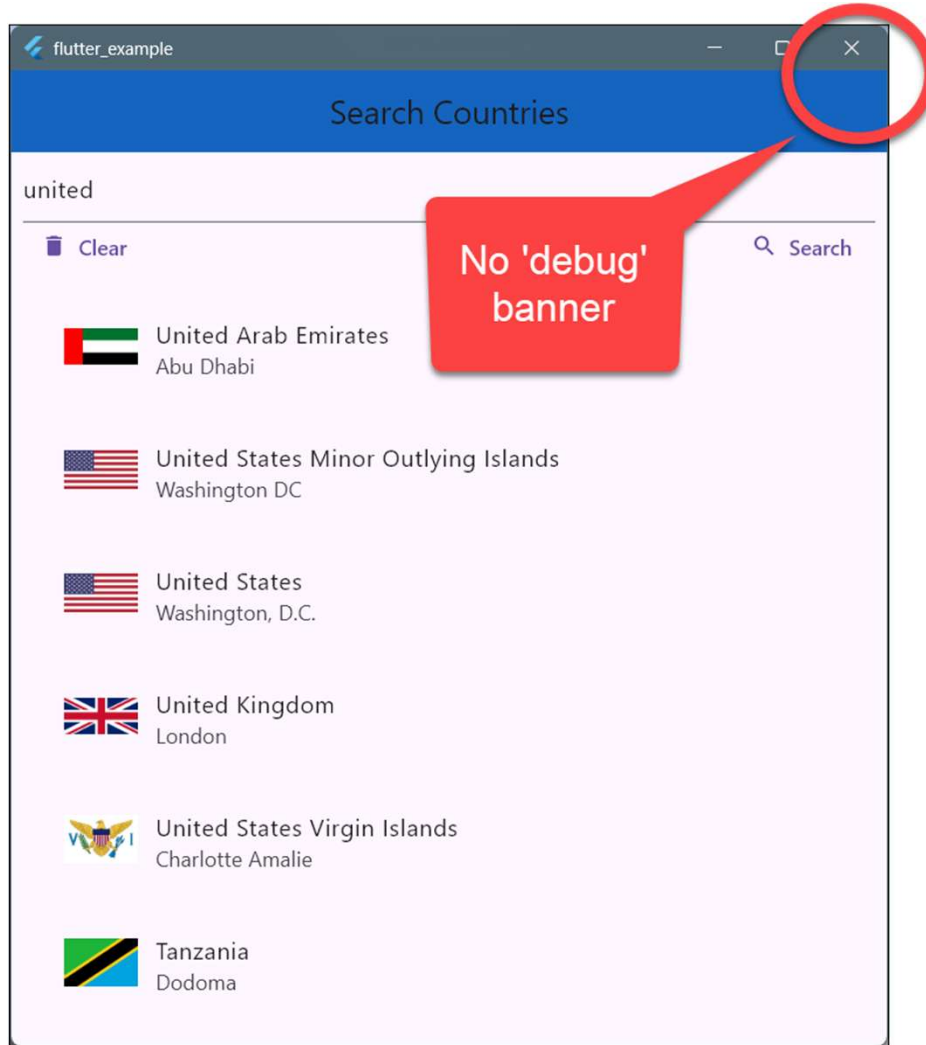
Approach for deployment

- Copy the **entire Release folder** to network drive or Artifactory, or:
 - Create a ZIP of it and extract it where needed
 - Write a simple script or installer that does the copy/deployment



Copied complete /Release folder to (shared) network location

Results



Sample PS script to automate build & deploy

```
# build_and_deploy.ps1. Easily build and deploy a Flutter-windows application
# from the command line in PowerShell.

# Set variables
$projectRoot = Split-Path -Parent $MyInvocation.MyCommand.Definition
$buildPath = Join-Path $projectRoot "build\windows\x64\runner\Release"

# Replace `your-network-drive` with actual UNC-path
$targetPath = "\\your-network-drive\path\to\app-deploy"

# Step 1: Build the app
Write-Host "Building Flutter Windows app..."
flutter build windows

# Step 2: Ensure target exists
if (!(Test-Path -Path $targetPath)) {
    Write-Host "Creating target directory at $targetPath"
    New-Item -ItemType Directory -Path $targetPath | Out-Null
}

# Step 3: Copy files
Write-Host "Copying release build to target..."
Copy-Item -Path "$buildPath\*" -Destination $targetPath -Recurse -Force

Write-Host "Done. App deployed to $targetPath"
```



Usage

- Save the script as `build_and_deploy.ps1` in Flutter project root.
- Edit `$targetPath` to point to your actual network path.
- Run it from PowerShell
 - `./build_and_deploy.ps1`



Building for other platforms

- Make sure to have the **platform prerequisites** installed
 - **Android**: Android SDK, correct PATH, etc
 - **iOS**: Xcode, command line tools, etc.
 - **Web**: fonts, assets, etc.
- Otherwise: errors

```
~\Desktop\flutter_example
flutter build appbundle
Downloading android-arm-profile/windows-x64 tools... 542ms
Downloading android-arm-release/windows-x64 tools... 348ms
Downloading android-arm64-profile/windows-x64 tools... 395ms
Downloading android-arm64-release/windows-x64 tools... 366ms
Downloading android-x64-profile/windows-x64 tools... 386ms
Downloading android-x64-release/windows-x64 tools... 365ms

[!] Your app is using an unsupported Gradle project. To fix this problem, create a new project
<app-directory>` and then move the dart code, assets and pubspec.yaml to the new project.
```

Other platforms - more



- Android

- Add launcher icon, sign the app, shrink app with R8, create App Manifest file, Build the app for release, Publish to Google Play Store
- <https://docs.flutter.dev/deployment/android>

- iOS

- Create app outline in App Store Connect, register Bundle ID, prepare app in Xcode, add App icon, launch image, build Archive from Xcode, create and upload app bundle
- <https://docs.flutter.dev/deployment/ios>

Workshop



- Use your own app, or use one of the example apps
- Create a **distribution build** for your platform
- Follow the steps described in the Flutter Docs, **Deployment** section

