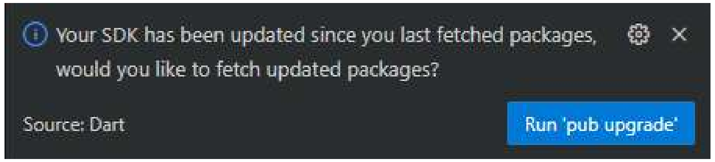
Build project that was created with Flutter 3.24.3 in Flutter 3.32.6

On the following pages we explain step-by-step how to migrate a project, that was created with Flutter version **3.24.3** (the one we installed in Sep. 2024) to Flutter **3.32.6**, which we have installed in July 2025.

As an example we used branch gs_ideas in the repository
https://github.com/fdg2425/bubbleTrouble_flutter_navalnorth_forked

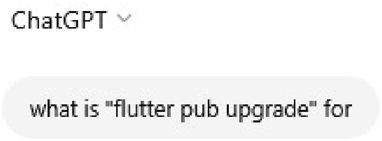
Open the project

When you had the project already on your PC and open it after Flutter was upgraded, you are asked:



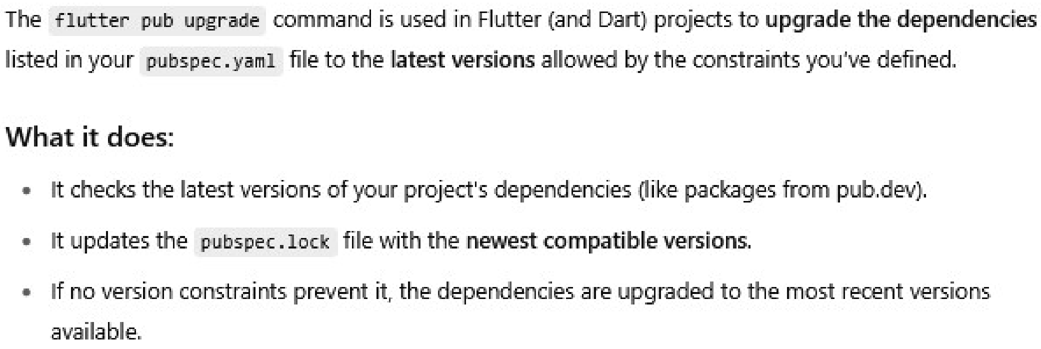
ⓘ Your SDK has been updated since you last fetched packages, would you like to fetch updated packages?

Source: Dart



ChatGPT

what is "flutter pub upgrade" for

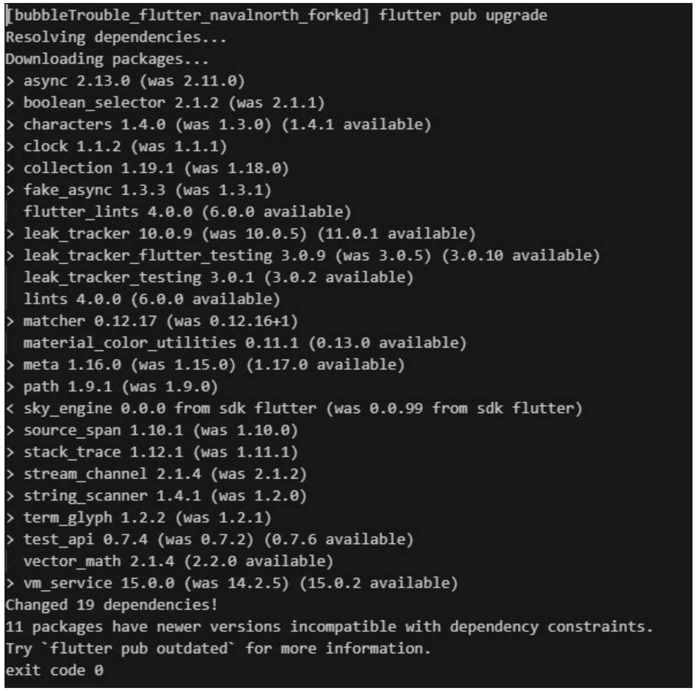


The `flutter pub upgrade` command is used in Flutter (and Dart) projects to **upgrade the dependencies** listed in your `pubspec.yaml` file to the **latest versions** allowed by the constraints you've defined.

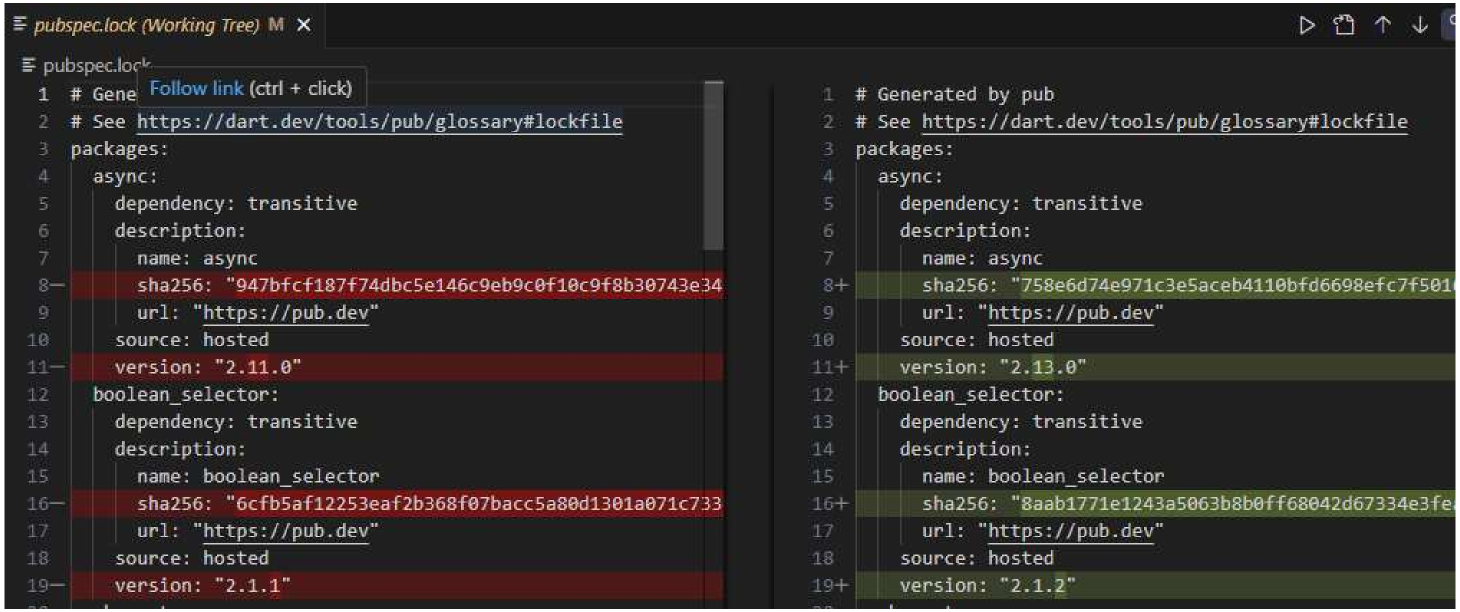
What it does:

- It checks the latest versions of your project's dependencies (like packages from pub.dev).
- It updates the `pubspec.lock` file with the **newest compatible versions.**
- If no version constraints prevent it, the dependencies are upgraded to the most recent versions available.

Executing “flutter pub upgrade”



```
[bubbleTrouble_flutter_navalnorth_forked] flutter pub upgrade
Resolving dependencies...
Downloading packages...
> async 2.13.0 (was 2.11.0)
> boolean_selector 2.1.2 (was 2.1.1)
> characters 1.4.0 (was 1.3.0) (1.4.1 available)
> clock 1.1.2 (was 1.1.1)
> collection 1.19.1 (was 1.18.0)
> fake_async 1.3.3 (was 1.3.1)
  flutter_lints 4.0.0 (6.0.0 available)
> leak_tracker 10.0.9 (was 10.0.5) (11.0.1 available)
> leak_tracker_flutter_testing 3.0.9 (was 3.0.5) (3.0.10 available)
  leak_tracker_testing 3.0.1 (3.0.2 available)
  lints 4.0.0 (6.0.0 available)
> matcher 0.12.17 (was 0.12.16+1)
  material_color_utilities 0.11.1 (0.13.0 available)
> meta 1.16.0 (was 1.15.0) (1.17.0 available)
> path 1.9.1 (was 1.9.0)
< sky_engine 0.0.0 from sdk flutter (was 0.0.99 from sdk flutter)
> source_span 1.10.1 (was 1.10.0)
> stack_trace 1.12.1 (was 1.11.1)
> stream_channel 2.1.4 (was 2.1.2)
> string_scanner 1.4.1 (was 1.2.0)
> term_glyph 1.2.2 (was 1.2.1)
> test_api 0.7.4 (was 0.7.2) (0.7.6 available)
  vector_math 2.1.4 (2.2.0 available)
> vm_service 15.0.0 (was 14.2.5) (15.0.2 available)
Changed 19 dependencies!
11 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
exit code 0
```

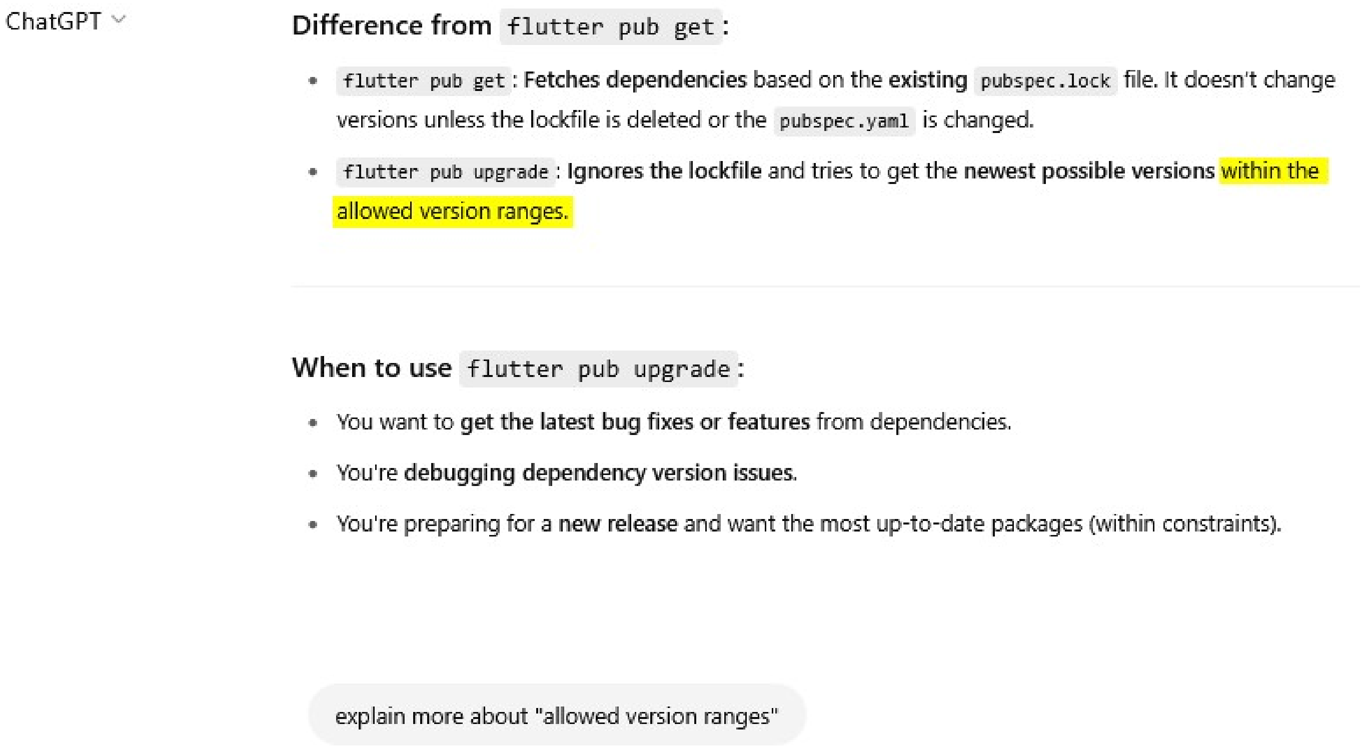


```
  # Gene
  # See https://dart.dev/tools/pub/glossary#lockfile
  packages:
    async:
      dependency: transitive
      description:
        name: async
-       sha256: "947bfcf187f74dbc5e146c9eb9c0f10c9f8b30743e34
        url: "https://pub.dev"
      source: hosted
-     version: "2.11.0"
    boolean_selector:
      dependency: transitive
      description:
        name: boolean_selector
-       sha256: "6cfb5af12253eaf2b368f07bacc5a80d1301a071c733
        url: "https://pub.dev"
      source: hosted
-     version: "2.1.1"
```

```
  # Generated by pub
  # See https://dart.dev/tools/pub/glossary#lockfile
  packages:
    async:
      dependency: transitive
      description:
        name: async
+       sha256: "758e6d74e971c3e5aceb4110bfd6698efc7f501
        url: "https://pub.dev"
      source: hosted
+     version: "2.13.0"
    boolean_selector:
      dependency: transitive
      description:
        name: boolean_selector
+       sha256: "8aab1771e1243a5063b8b0ff68042d67334e3fe
        url: "https://pub.dev"
      source: hosted
+     version: "2.1.2"
```

pubspec.lock keeps track, which versions of the dependent packages were last used in the project.

Difference “flutter pub get” and “flutter pub upgrade”



Difference from `flutter pub get`:

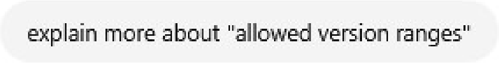
- `flutter pub get`: **Fetches dependencies** based on the **existing** `pubspec.lock` file. It doesn't change versions unless the lockfile is deleted or the `pubspec.yaml` is changed.
- `flutter pub upgrade`: **Ignores the lockfile** and tries to get the **newest possible versions** within the allowed version ranges.

When to use `flutter pub upgrade`:

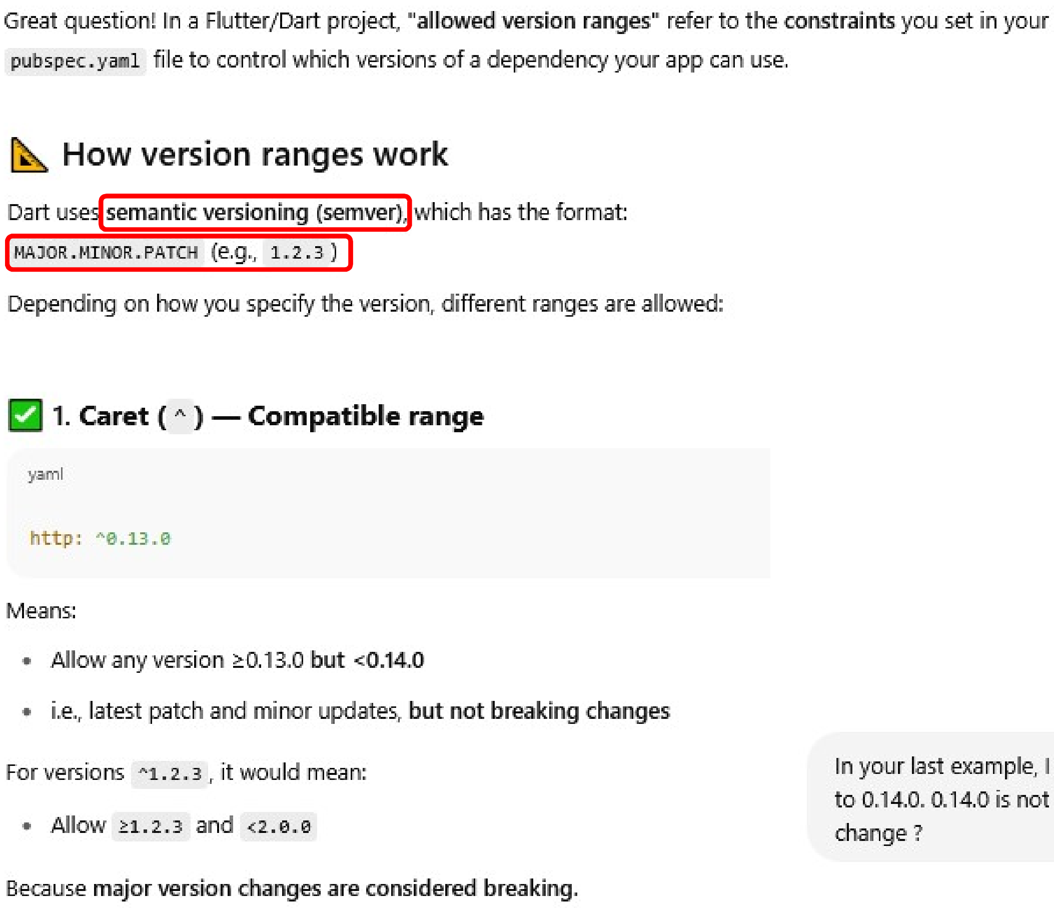
- You want to **get the latest bug fixes or features** from dependencies.
- You're **debugging dependency version issues.**
- You're preparing for a **new release** and want the most up-to-date packages (within constraints).

explain more about "allowed version ranges"

Semantic versioning and version constraints



explain more about "allowed version ranges"



Great question! In a Flutter/Dart project, **"allowed version ranges"** refer to the **constraints** you set in your `pubspec.yaml` file to control which versions of a dependency your app can use.

📐 How version ranges work

Dart uses **semantic versioning (semver)**, which has the format:
`MAJOR.MINOR.PATCH` (e.g., `1.2.3`)

Depending on how you specify the version, different ranges are allowed:

✅ 1. Caret (`^`) — Compatible range

yaml

```yaml
http: ^0.13.0
```

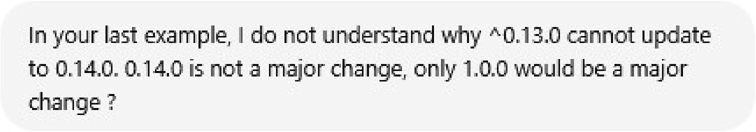
Means:

- Allow any version ≥0.13.0 **but <0.14.0**
- i.e., latest patch and minor updates, **but not breaking changes**

For versions `^1.2.3`, it would mean:

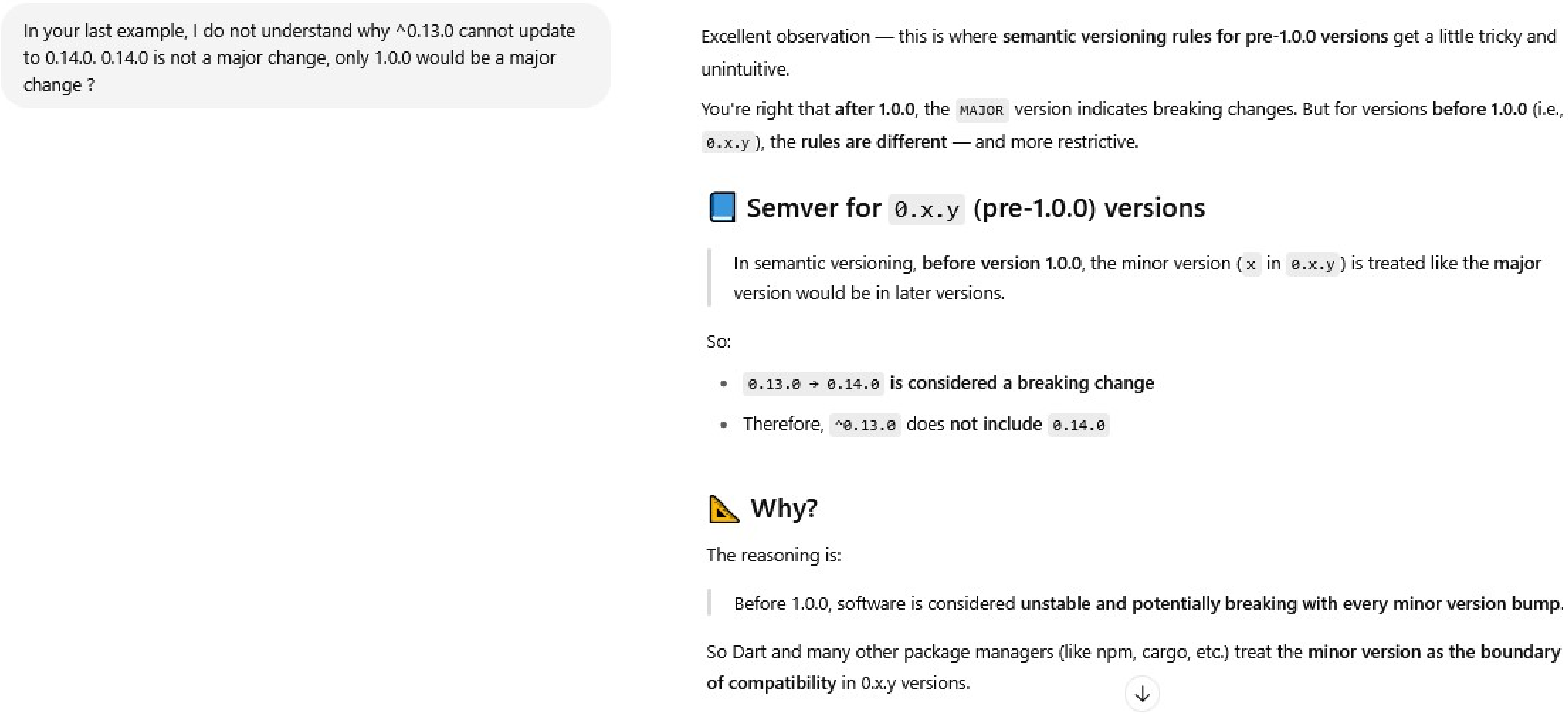
- Allow `≥1.2.3` and `<2.0.0`

Because **major version changes are considered breaking.**



In your last example, I do not understand why ^0.13.0 cannot update to 0.14.0. 0.14.0 is not a major change, only 1.0.0 would be a major change ?

Version constraints for versions below 1.0.0



In your last example, I do not understand why ^0.13.0 cannot update to 0.14.0. 0.14.0 is not a major change, only 1.0.0 would be a major change ?

Excellent observation — this is where **semantic versioning rules for pre-1.0.0 versions** get a little tricky and unintuitive.

You're right that **after 1.0.0**, the `MAJOR` version indicates breaking changes. But for versions **before 1.0.0** (i.e., `0.x.y`), the **rules are different** — and more restrictive.

📘 Semver for `0.x.y` (pre-1.0.0) versions

> In semantic versioning, **before version 1.0.0**, the minor version (`x` in `0.x.y`) is treated like the **major** version would be in later versions.

So:

- `0.13.0 → 0.14.0` **is considered a breaking change**
- Therefore, `^0.13.0` does **not include** `0.14.0`

📐 Why?

The reasoning is:

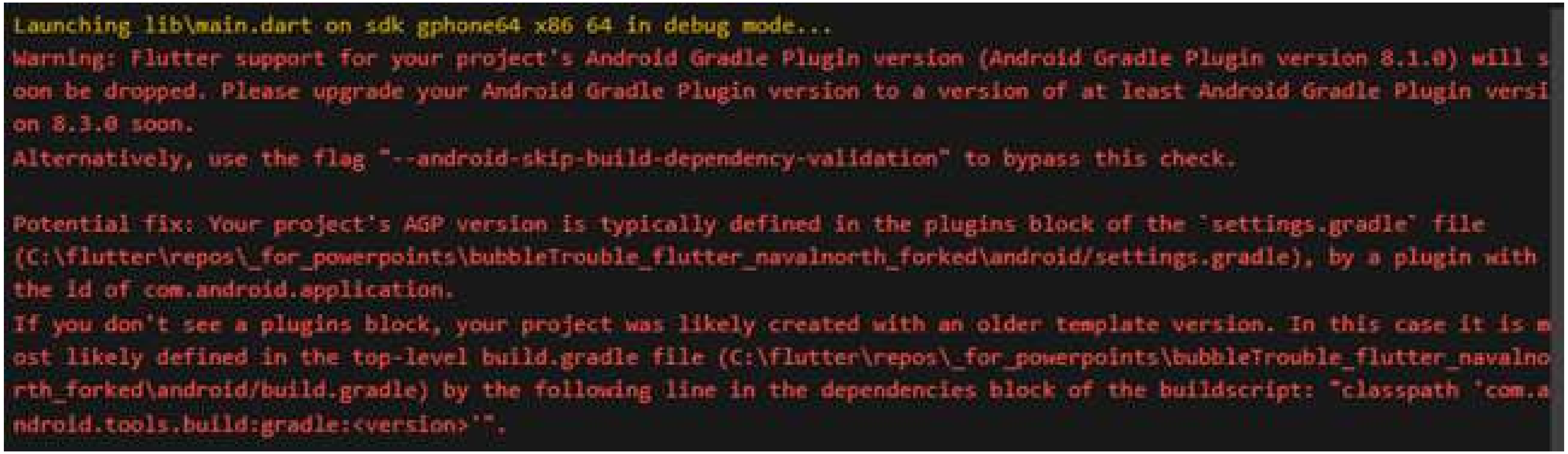
> Before 1.0.0, software is considered **unstable and potentially breaking with every minor version bump**.

So Dart and many other package managers (like npm, cargo, etc.) treat the **minor version as the boundary of compatibility** in 0.x.y versions.

Warning during the first build for Android

The first build for Chrome under the new flutter version succeeds without issues.

But when building for Android we get:

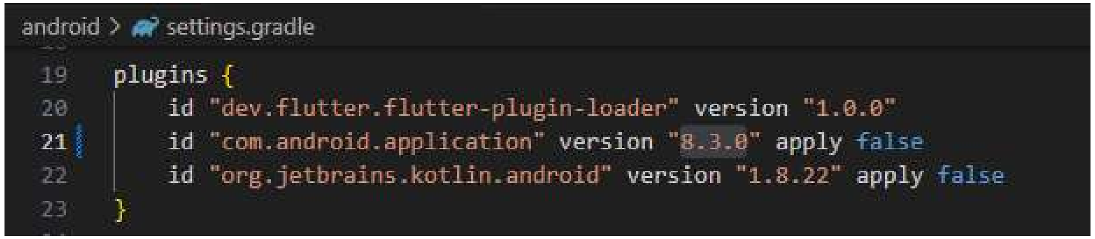


```
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Warning: Flutter support for your project's Android Gradle Plugin version (Android Gradle Plugin version 8.1.0) will s
oon be dropped. Please upgrade your Android Gradle Plugin version to a version of at least Android Gradle Plugin versi
on 8.3.0 soon.
Alternatively, use the flag "--android-skip-build-dependency-validation" to bypass this check.

Potential fix: Your project's AGP version is typically defined in the plugins block of the `settings.gradle` file
(C:\flutter\repos\_for_powerpoints\bubbleTrouble_flutter_navalnorth_forked\android/settings.gradle), by a plugin with
the id of com.android.application.
If you don't see a plugins block, your project was likely created with an older template version. In this case it is m
ost likely defined in the top-level build.gradle file (C:\flutter\repos\_for_powerpoints\bubbleTrouble_flutter_navalno
rth_forked\android/build.gradle) by the following line in the dependencies block of the buildscript: "classpath 'com.a
ndroid.tools.build:gradle:<version>'".
```

This is only a warning. The build continues and the app successfully starts on Android.

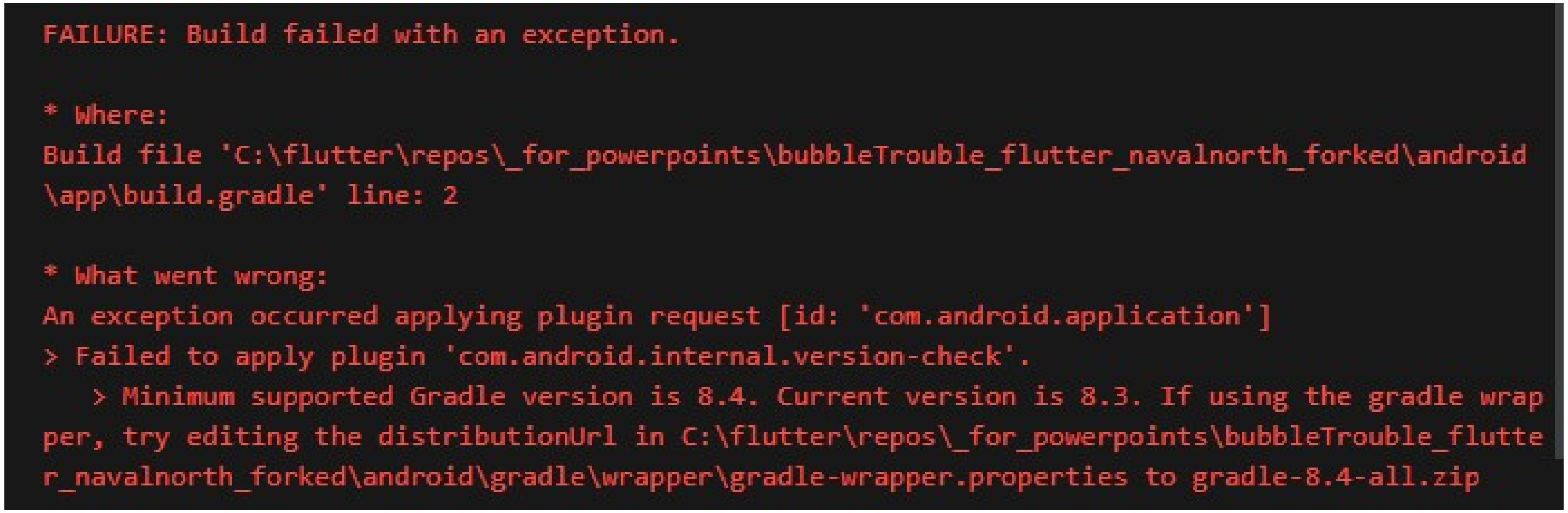
Using the recommended AGP 8.3.0

“AGP” is short for
“Android Gradle Plugin”



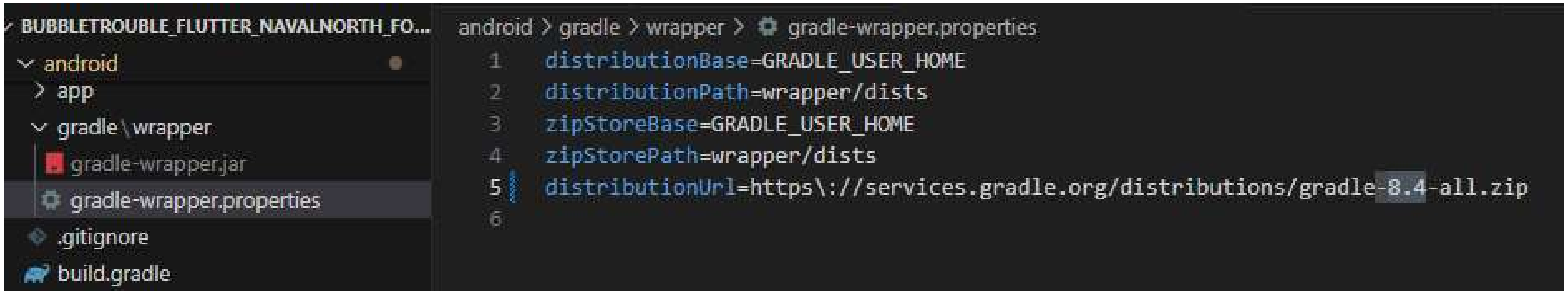
```
plugins {
    id "dev.flutter.flutter-plugin-loader" version "1.0.0"
    id "com.android.application" version "8.3.0" apply false
    id "org.jetbrains.kotlin.android" version "1.8.22" apply false
}
```

This is a
FAILURE,
thus it breaks
the build!

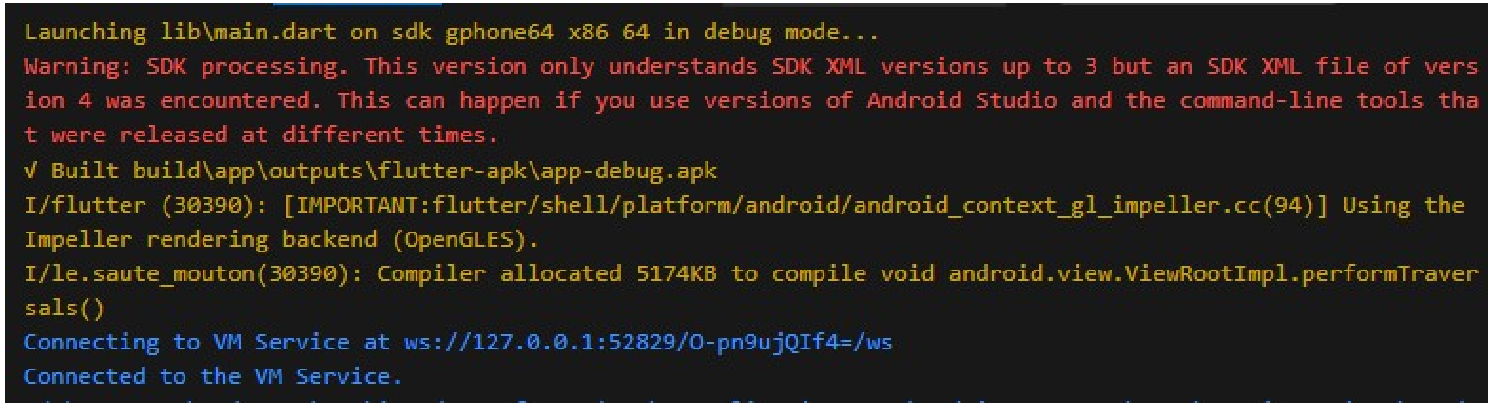


```
FAILURE: Build failed with an exception.

* Where:
Build file 'C:\flutter\repos\_for_powerpoints\bubbleTrouble_flutter_navalnorth_forked\android
\app\build.gradle' line: 2

* What went wrong:
An exception occurred applying plugin request [id: 'com.android.application']
> Failed to apply plugin 'com.android.internal.version-check'.
   > Minimum supported Gradle version is 8.4. Current version is 8.3. If using the gradle wrap
per, try editing the distributionUrl in C:\flutter\repos\_for_powerpoints\bubbleTrouble_flutte
r_navalnorth_forked\android\gradle\wrapper\gradle-wrapper.properties to gradle-8.4-all.zip
```

Setting Gradle version to 8.4.0


BUBBLETROUBLE_FLUTTER_NAVALNORTH_FO...
android
app
gradle \ wrapper
gradle-wrapper.jar
gradle-wrapper.properties
.gitignore
build.gradle
android > gradle > wrapper > gradle-wrapper.properties

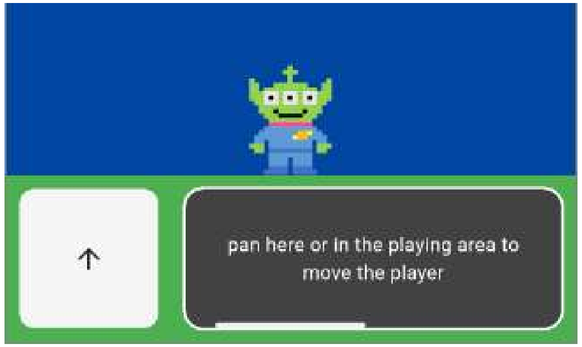
```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-8.4-all.zip
```



```
Launching lib\main.dart on sdk gphone64 x86 64 in debug mode...
Warning: SDK processing. This version only understands SDK XML versions up to 3 but an SDK XML file of vers
ion 4 was encountered. This can happen if you use versions of Android Studio and the command-line tools tha
t were released at different times.
√ Built build\app\outputs\flutter-apk\app-debug.apk
I/flutter (30390): [IMPORTANT:flutter/shell/platform/android/android_context_gl_impeller.cc(94)] Using the
Impeller rendering backend (OpenGLES).
I/le.saute_mouton(30390): Compiler allocated 5174KB to compile void android.view.ViewRootImpl.performTraver
sals()
Connecting to VM Service at ws://127.0.0.1:52829/O-pn9ujQIf4=/ws
Connected to the VM Service.
```

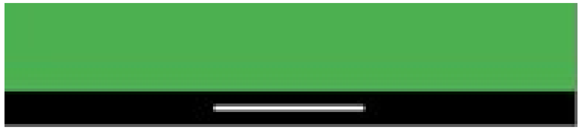
Now there is only one warning left, and this warning no longer appears on subsequent builds.

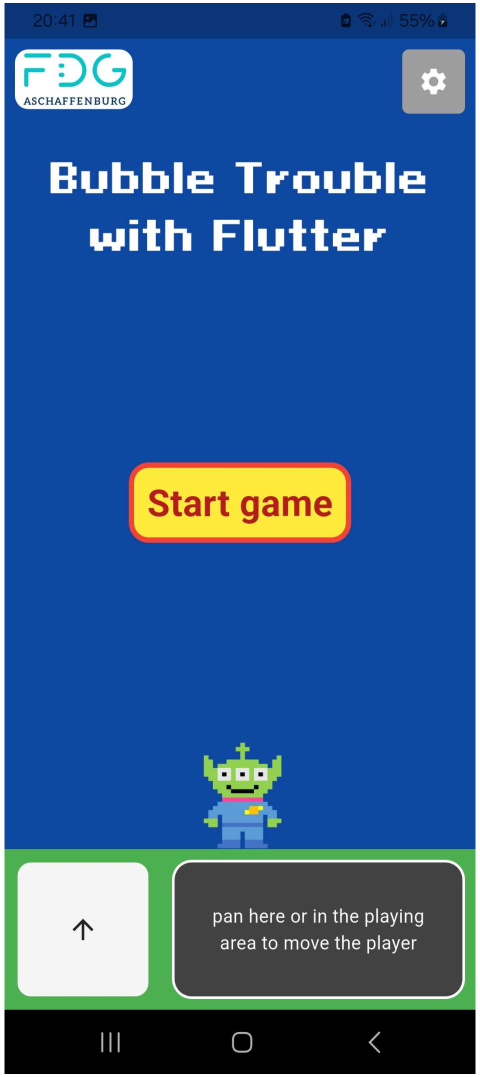
UI changes in the new Flutter version



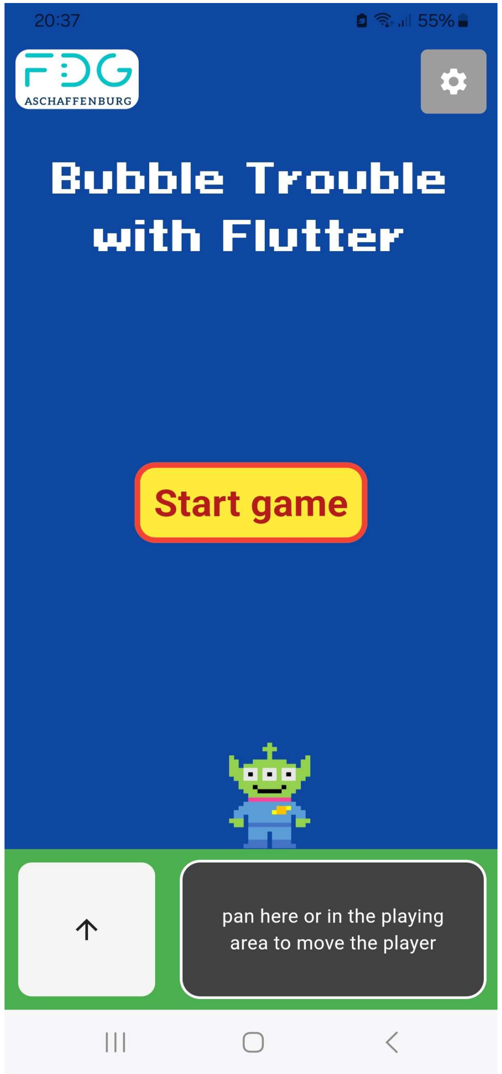
“Strange” white line in Android Emulator

old:



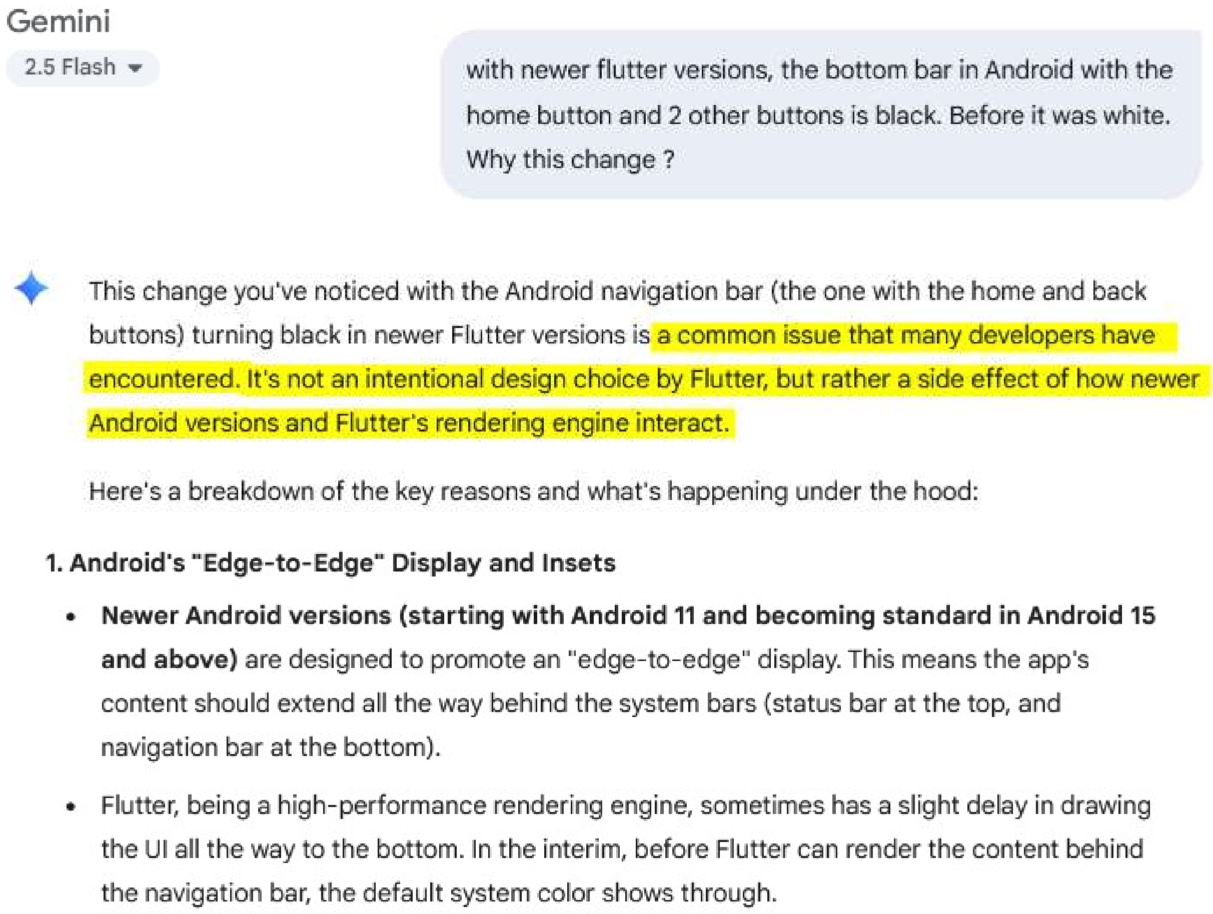

20:41
55%
FDG
ASCHAFFENBURG
Bubble Trouble with Flutter
Start game
↑
pan here or in the playing area to move the player

Created with new Flutter


20:37
55%
FDG
ASCHAFFENBURG
Bubble Trouble with Flutter
Start game
↑
pan here or in the playing area to move the player

Created with old Flutter

Explanation in Gemini



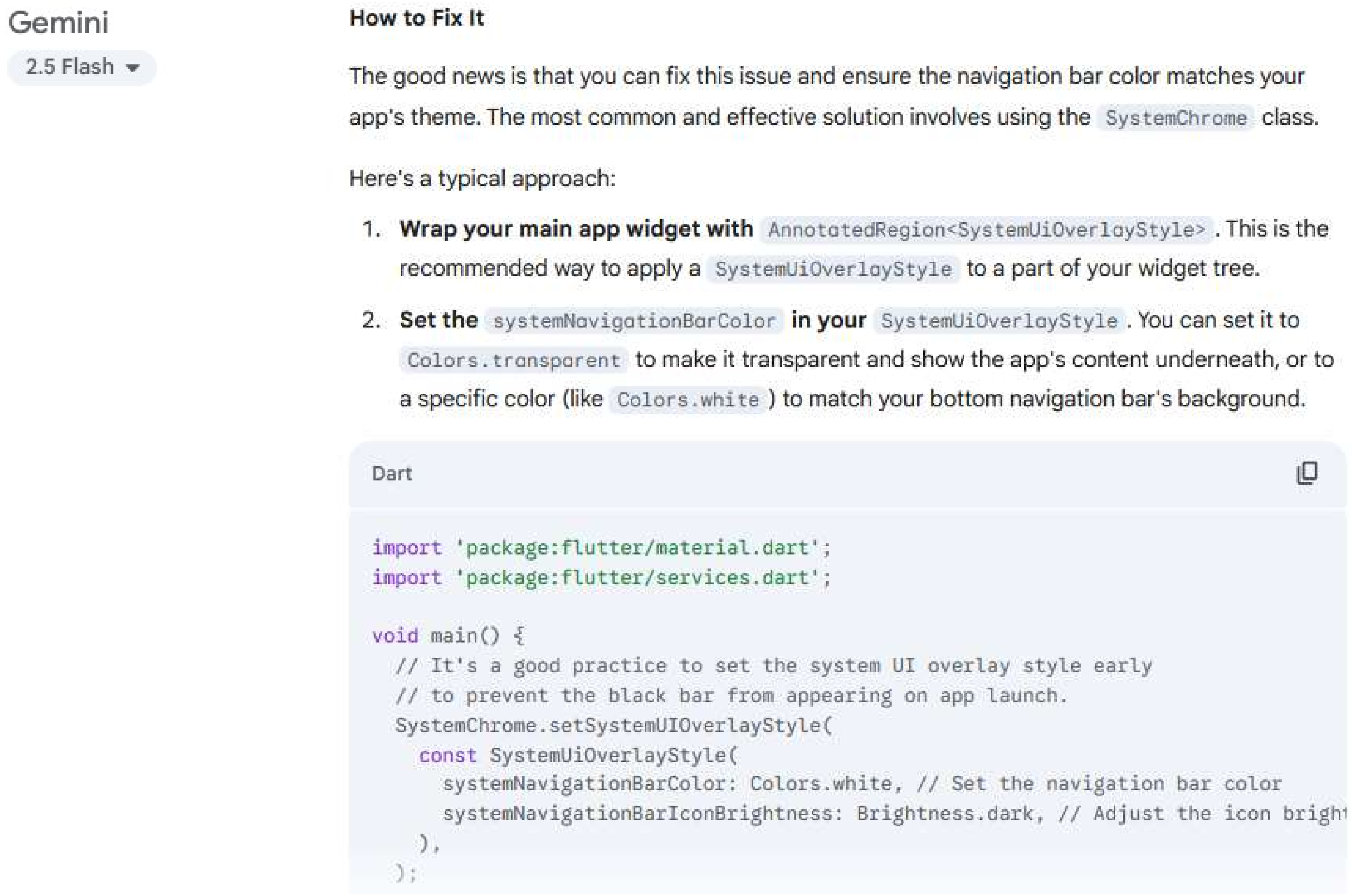
with newer flutter versions, the bottom bar in Android with the home button and 2 other buttons is black. Before it was white. Why this change ?

This change you've noticed with the Android navigation bar (the one with the home and back buttons) turning black in newer Flutter versions is a common issue that many developers have encountered. It's not an intentional design choice by Flutter, but rather a side effect of how newer Android versions and Flutter's rendering engine interact.

Here's a breakdown of the key reasons and what's happening under the hood:

1. **Android's "Edge-to-Edge" Display and Insets**
   - **Newer Android versions (starting with Android 11 and becoming standard in Android 15 and above)** are designed to promote an "edge-to-edge" display. This means the app's content should extend all the way behind the system bars (status bar at the top, and navigation bar at the bottom).
   - Flutter, being a high-performance rendering engine, sometimes has a slight delay in drawing the UI all the way to the bottom. In the interim, before Flutter can render the content behind the navigation bar, the default system color shows through.

Possible fix according to Gemini (not yet implemented)



How to Fix It

The good news is that you can fix this issue and ensure the navigation bar color matches your app's theme. The most common and effective solution involves using the `SystemChrome` class.

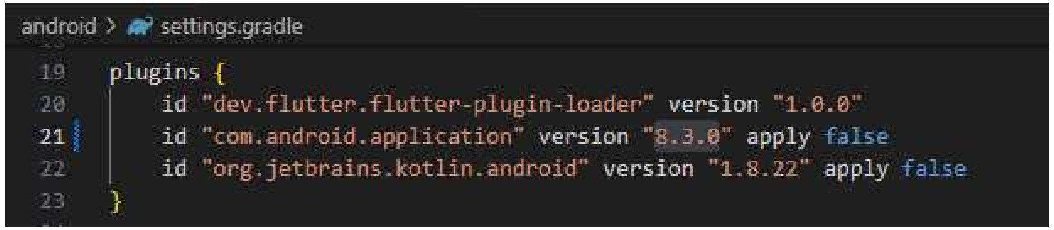
Here's a typical approach:

1. **Wrap your main app widget with** `AnnotatedRegion<SystemUiOverlayStyle>`. This is the recommended way to apply a `SystemUiOverlayStyle` to a part of your widget tree.
2. **Set the** `systemNavigationBarColor` **in your** `SystemUiOverlayStyle`. You can set it to `Colors.transparent` to make it transparent and show the app's content underneath, or to a specific color (like `Colors.white`) to match your bottom navigation bar's background.

```dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

void main() {
  // It's a good practice to set the system UI overlay style early
  // to prevent the black bar from appearing on app launch.
  SystemChrome.setSystemUIOverlayStyle(
    const SystemUiOverlayStyle(
      systemNavigationBarColor: Colors.white, // Set the navigation bar color
      systemNavigationBarIconBrightness: Brightness.dark, // Adjust the icon brigh
    ),
  );
```

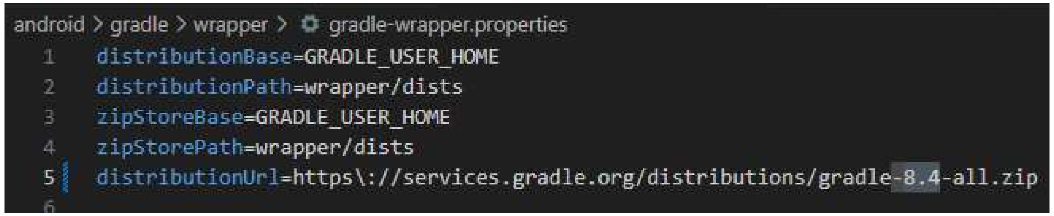
Comparison with a project created with Flutter 3.32.6

Our changes described in the last pages:



```
plugins {
    id "dev.flutter.flutter-plugin-loader" version "1.0.0"
    id "com.android.application" version "8.3.0" apply false
    id "org.jetbrains.kotlin.android" version "1.8.22" apply false
}
```

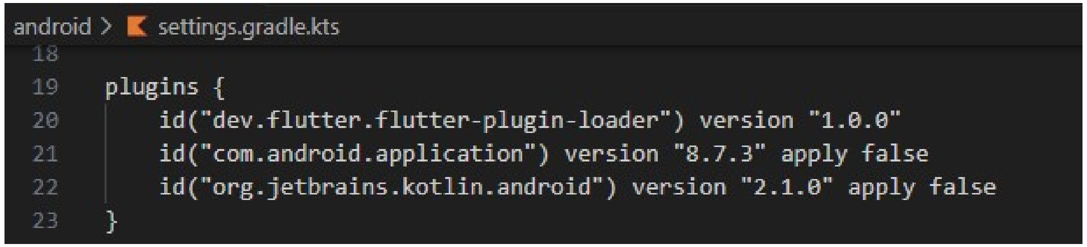
Old entry from Flutter 3.24.3 was “**8.1.0**” here.



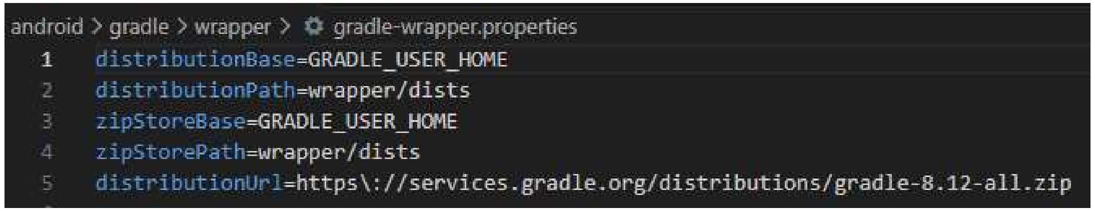
```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-8.4-all.zip
```

Old entry from Flutter 3.24.3 was “gradle-**8.3**-all.zip”.

Settings in lux_meter_2025, which has been created with new Flutter 3.32.6:



```

plugins {
    id("dev.flutter.flutter-plugin-loader") version "1.0.0"
    id("com.android.application") version "8.7.3" apply false
    id("org.jetbrains.kotlin.android") version "2.1.0" apply false
}
```



```
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-8.12-all.zip
```