EcomLabs

STEP-BY-STEP GUIDE

Migrating to the React Native New Architecture

The migration checklist from React Native 0.81 or Expo SDK 54 up to the current release, where to turn the New Architecture on, a Node script that sorts your native libraries, and a table to audit them one by one.

FOR	React Native or Expo apps still on the Legacy Architecture, or not sure which one they use
TESTED WITH	Node 22.22.3 · audit script run against a test folder · no device build
UPDATED	November 19, 2026
DOCS	reactnative.dev/blog/2025/10/08/react-native-0.82 · docs.expo.dev/guides/new-architecture
ARTICLE	ecomlabs.dev/g/arquitectura-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	Where you are and where you are going
Step 01	Move to 0.81 or SDK 54
Step 02	Turn the New Architecture on
Step 03	Audit your native libraries
Step 04	When a library is not compatible
Step 05	Upgrade to the current release
When you finish	Checklist

BEFORE YOU START

Where you are and where you are going

Since React Native 0.82 (October 8, 2025) the New Architecture is the only one: `newArchEnabled=false` on Android and `RCT_NEW_ARCH_ENABLED=0` on iOS are ignored. 0.81 and Expo SDK 54 are the last versions that allow the Legacy Architecture. As of November 19, 2026, the current release is 0.88, and Expo SDK 58 ships it. It has been on by default since 0.76: if nobody turned it off (step 02), you already use it.

On 0.80 or older, start at step 01. On 0.81 or SDK 54 with the Legacy Architecture, start at step 02. Already on the New Architecture, go to step 05, and come back to step 03 when something breaks.

> **CAUTION**
> Before you change anything, decide how you get back: keep the last working build, write down its version and keep the branch ready. An app in the stores does not roll back with one click.

STEP 01

Move to 0.81 or SDK 54

Do this step without changing architecture. Expo recommends upgrading one SDK at a time; if you are on SDK 52, go to 53 first, then 54.

```
# Expo project: repeat for each SDK up to 54
npm install expo@^54.0.0
npx expo install --fix
npx expo-doctor
```

The first upgrades `expo`, the second aligns the other dependencies with that SDK, and the third checks for common problems: fix them before you continue. Without Expo, use the Upgrade Helper (`react-native-community.github.io/upgrade-helper`) to see the file changes between your version and 0.81, and install `react-native` and `react` at the versions it shows. Test the whole app on this version. If something breaks here, the cause is the version, not the architecture.

STEP 02

Turn the New Architecture on

The Expo guide documents three places where it gets turned off. Search for them from the project root:

```
grep -rn "newArchEnabled" app.json android/gradle.properties ios/Podfile.properties.json
grep -rn "RCT_NEW_ARCH_ENABLED" ios/Podfile package.json
```

FILE	OFF	ON
`app.json` (SDK 54)	`"newArchEnabled": false`	`"newArchEnabled": true` or no line
`android/gradle.properties`	`newArchEnabled=false`	`newArchEnabled=true`
`ios/Podfile.properties.json`	`"newArchEnabled": "false"`	`"newArchEnabled": "true"`
Scripts or `Podfile`	`RCT_NEW_ARCH_ENABLED=0`	Remove the variable and reinstall pods

If you use `app.config.ts`, search there too. Then create a fresh development build (`npx expo run:android` and `npx expo run:ios`, or your cloud build). Expo Go cannot test your own native libraries.

STEP 03

Audit your native libraries

JavaScript-only libraries are not affected. Libraries that ship `ios/`, `android/` or a `.podspec` are. This script sorts them by reading your `node_modules`, offline. Run it after `npm install`:

audit-native-deps.mjs

```
// audit-native-deps.mjs: list the dependencies that ship native code and what they declare.
// Usage: node audit-native-deps.mjs [path/to/app]   (run after npm install)
import { readFileSync, existsSync, readdirSync } from "node:fs";
import { join } from "node:path";

const root = process.argv[2] ?? ".";
const app = JSON.parse(readFileSync(join(root, "package.json"), "utf8"));
const names = Object.keys({ ...app.dependencies, ...app.devDependencies }).sort();

const rows = [];
for (const name of names) {
  const dir = join(root, "node_modules", name);
  if (!existsSync(join(dir, "package.json"))) {
    rows.push([name, "?", "not installed"]);
    continue;
  }
  const pkg = JSON.parse(readFileSync(join(dir, "package.json"), "utf8"));
  const files = readdirSync(dir);
  const native = files.includes("ios") || files.includes("android") || files.some((f) =>
f.endsWith(".podspec"));
  if (!native) continue; // JavaScript only: the architecture does not affect it
  const verdict = pkg.codegenConfig
    ? "codegenConfig: ships New Architecture specs"
    : files.includes("expo-module.config.json")
      ? "Expo module: supports the New Architecture"
      : "CHECK: native code, no codegenConfig";
  rows.push([name, pkg.version, verdict]);
}

for (const [name, version, verdict] of rows) console.log(`${name.padEnd(34)} ${String(version).padEnd(9)}
```

```
${verdict}`);
const check = rows.filter((r) => r[2].startsWith("CHECK") || r[1] === "?").length;
console.log(`\n${rows.length} native or missing, ${check} to check by hand`);
process.exit(check ? 1 : 0);
```

`codegenConfig` means the library ships specs for TurboModules or Fabric; `Expo module` means it is an Expo module, and those support the New Architecture. Test them in the build anyway. `CHECK` is native code with no spec: it may run through the interop layer or break, and it goes in your audit table. The script exits with code 1 when there is something to review, so you can put it in CI. We ran it with Node 22 against a test folder of invented packages; it is a filter, not a verdict.

Two more sources for the CHECK list

On React Native Directory (`reactnative.directory`) each library carries a `newArchitecture` field, and `npx expo-doctor` checks your dependencies against that data. If you already reviewed one the directory does not know, exclude it in `package.json` under `expo.doctor.reactNativeDirectoryCheck.exclude`.

Audit table

Keep a table like this in your repository, one row per CHECK library. The rows are illustrative:

LIBRARY	VERSION	DIRECTORY	SCREEN TESTED	RESULT	ACTION
old-native-lib	0.9.3	no data	Profile, photo upload	Crashes on open	Replace
@acme/maps	1.4.0	yes	Store map	Works	Update to latest

Add one row per library. A row closes only after you opened the screen that uses it in the development build, on iOS and on Android.

STEP 04

When a library is not compatible

In this order, following the Expo guide:

1. **Update it** to its latest version. Many have added support.
2. **Isolate it**: create a branch, remove the failing libraries until the app starts, and measure how much you depend on each one.
3. **Replace it** with a compatible one. Expo suggests, for example, `expo-location` instead of `react-native-geolocation-service`.
4. **Report it** with an issue or a pull request on its repository, with a minimal reproduction.
5. **If you own it, migrate it**: write its spec in TypeScript with the `Native` prefix (for example, `NativeLocalStorage.ts`), add `codegenConfig` to its `package.json` and regenerate.

STEP 05

Upgrade to the current release

Staying on SDK 54 with the Legacy Architecture is a pause, not a plan. With the app stable on the New Architecture, move up one SDK at a time (55, 56, 57, 58): repeat the step 01 commands with the new number and run the step 03 script again. SDK 55 ships React Native 0.83 and removes `newArchEnabled` from `app.json`; SDK 58 ships 0.88. Without Expo, use the Upgrade Helper one version at a time.

WHEN YOU FINISH

Checklist

- [ ] A previous build is saved and there is a way back to it.
- [ ] The app runs stable on React Native 0.81 or SDK 54.
- [ ] No file sets `newArchEnabled` to `false` or `RCT_NEW_ARCH_ENABLED=0`.
- [ ] A fresh development build runs with the New Architecture on.
- [ ] The audit script ran and every CHECK row is in the table.
- [ ] Every screen that uses a native library was opened on iOS and Android.
- [ ] `npx expo-doctor` passes, or every exclusion has a reason.
- [ ] Every incompatible library is updated, replaced, reported or migrated.
- [ ] The upgrade continues one SDK at a time up to the current release.