

Android Studio Beginner Guide

by Deekay

A professional A-Z beginner guide for learning Android app development with Android Studio, Kotlin, and Jetpack Compose. Built for a fresh beginner who wants practical steps, clear explanations, diagrams, and a first Hello World project.

Created: 21 June 2026. Sources are summarized in beginner-friendly wording from Android Developers and Kotlin documentation.

Table of Contents

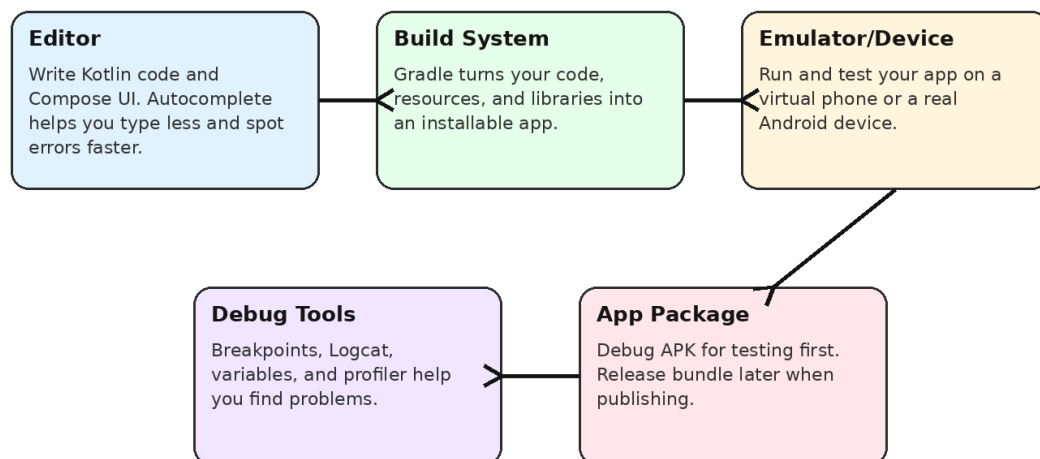
1. What Android development is
2. What Android Studio does
3. What to install and check
4. Kotlin basics you need first
5. Jetpack Compose basics
6. Project structure explained
7. Build, run, emulator, and real device
8. Hello World step-by-step
9. Debugging beginner errors
10. Permissions and privacy basics
11. App architecture basics
12. 30-day learning path
13. Best official tutorials
14. Sources

1. What Android Development Is

Android development means creating apps for Android phones, tablets, TVs, cars, watches, and other Android-powered devices. For a beginner, focus on normal phone apps first. Your first goal is not to build a massive app. Your first goal is to understand the loop: write code, build, run, test, fix, repeat.

Modern Android apps are commonly built with Kotlin and Jetpack Compose. Kotlin is the programming language. Jetpack Compose is the UI toolkit. Android Studio is the program on your laptop where you write, build, run, and debug the app.

What Android Studio Does



Beginner idea: code -> build -> run -> test -> fix -> repeat.

Android Studio Guide by Deekay

2. What Android Studio Does

Android Studio is the official Integrated Development Environment for Android app development. It is based on IntelliJ IDEA and adds Android-specific tools: project templates, an Android SDK manager, emulator manager, code editor, design tools, Gradle build integration, debugger, Logcat, profilers, and release tooling.

The main areas you must recognize

- Project window: shows files and folders.
- Editor: where you edit Kotlin, XML, Gradle, and other files.
- Run button: builds and installs your app on an emulator or real device.
- Build window: shows compile and Gradle errors.
- Logcat: shows app logs and crash messages.
- Device Manager: creates and starts Android emulators.
- SDK Manager: installs Android platform and tools.

Beginner tip: do not memorize every menu. Learn the parts you use every day: Project, Editor, Run, Build, Logcat, Device Manager.

3. What to Install and Check

You already installed Android Studio. Open it and make sure these pieces are ready: Android SDK, Android SDK Platform, Android Emulator, Platform Tools, and at least one Android system image for the emulator. Android Studio usually guides you during first launch.

Laptop checks

- Enough disk space. Android Studio, SDKs, Gradle caches, and emulators can use many GB.
- Virtualization enabled for emulator performance. On Windows this can involve BIOS/UEFI virtualization and Windows hypervisor settings.
- Internet access for Gradle sync because dependencies are downloaded from Google and Maven repositories.
- JDK is bundled with Android Studio; most beginners do not need to install a separate Java kit.

Recommended beginner setup

Start with a normal Empty Activity / Empty Compose Activity project. Use Kotlin. Use Compose. Use a Pixel emulator first, then test on your real phone once the app runs.

4. Kotlin Basics You Need First

Kotlin is Android's recommended language for modern Android development. As a beginner, learn enough Kotlin to read MainActivity.kt and write small functions. Do not try to master everything before building screens.

Core Kotlin concepts

- val means read-only value. var means value can change.
- fun creates a function.
- String, Int, Boolean, Double are common data types.
- if and when are used for decisions.
- for and while are loops.
- class and data class create your own types.
- null safety helps prevent common crashes by making nullable values explicit.
- Lists store multiple values.

Tiny Kotlin example

```
val appName = "Hello Deekay"
var clickCount = 0

fun greeting(name: String): String {
    return "Hello, $name"
}

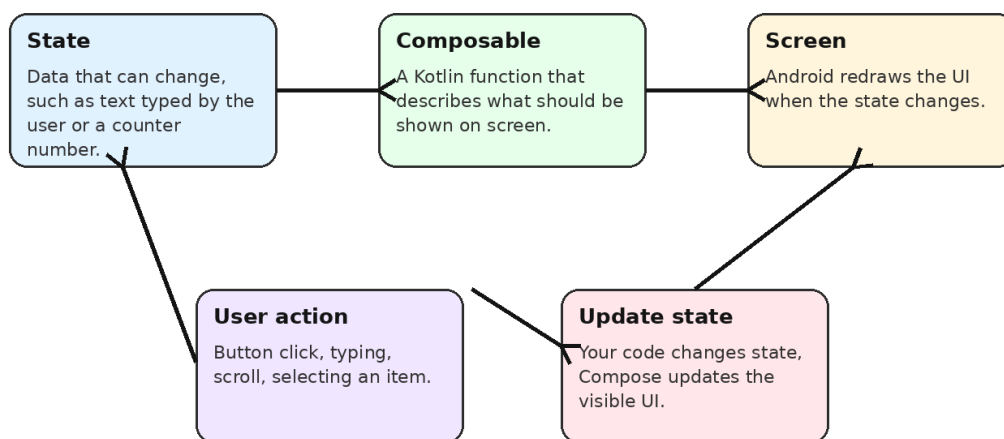
if (clickCount == 0) {
    println("Button not clicked yet")
} else {
    println("Button clicked")
}
```

Practice Kotlin in small pieces. If a line confuses you, rewrite it in a simpler way. Kotlin becomes easier once you build small apps with it.

5. Jetpack Compose Basics

Jetpack Compose is Android's modern toolkit for building native UI. Instead of editing XML layout files first, you write Kotlin functions that describe what the screen should look like. These functions are called composables.

How Jetpack Compose UI Works



Declarative UI means: describe the screen for the current state; Compose handles updates.

Android Studio Guide by Deekay

Important Compose words

- `@Composable`: tells Compose that a function draws UI.
- `Text()`: shows text.
- `Button()`: creates a clickable button.
- `Column()`: places items vertically.
- `Row()`: places items horizontally.
- `Modifier`: changes size, padding, alignment, background, and behavior.
- `remember` and `mutableStateOf`: store simple UI state.
- `@Preview`: shows a preview inside Android Studio.

Small Compose example

```
@Composable
fun HelloScreen() {
    var count by remember { mutableStateOf(0) }

    Column {
        Text("Hello Android Studio!")
        Text("Clicked: $count")
        Button(onClick = { count++ }) {
```

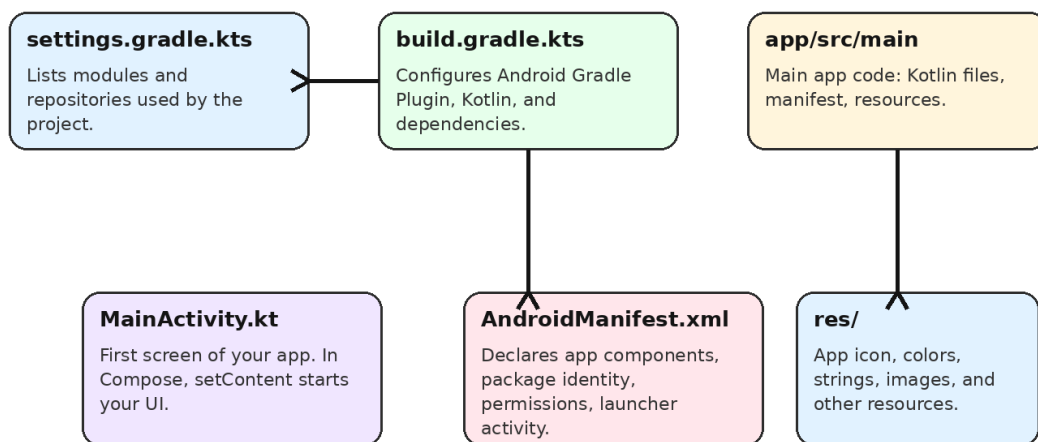
```
Text("Click Me")
}
}
}
```

Beginner meaning: the screen shows text and a button. When the button is clicked, count changes. Compose sees the change and redraws the text.

6. Project Structure Explained

A new Android project contains more than one file. Beginners often panic because there are many folders. You only need to understand the important ones first.

Android Project Structure



Open the whole project folder in Android Studio, not only MainActivity.kt.

Android Studio Guide by Deekay

Files you will touch often

- MainActivity.kt: your first app screen and entry point.
- app/build.gradle.kts: app settings and dependencies.
- AndroidManifest.xml: app declarations, launcher activity, permissions.
- res/: app resources such as icons, images, colors, and strings.
- settings.gradle.kts: declares modules like :app.

Rule: open the full project folder in Android Studio. Do not open only a single Kotlin file or Gradle cannot understand the project.

7. Build, Run, Emulator, and Real Device

To run an app, Android Studio builds the project, installs it on a selected device, and launches it. The device can be a virtual phone using Android Emulator or a real Android phone connected by USB/Wi-Fi debugging.

Using the emulator

- Open Device Manager.
- Create a virtual device.
- Choose a common phone such as Pixel.
- Download a recommended system image.
- Start the emulator.
- Press Run in Android Studio.

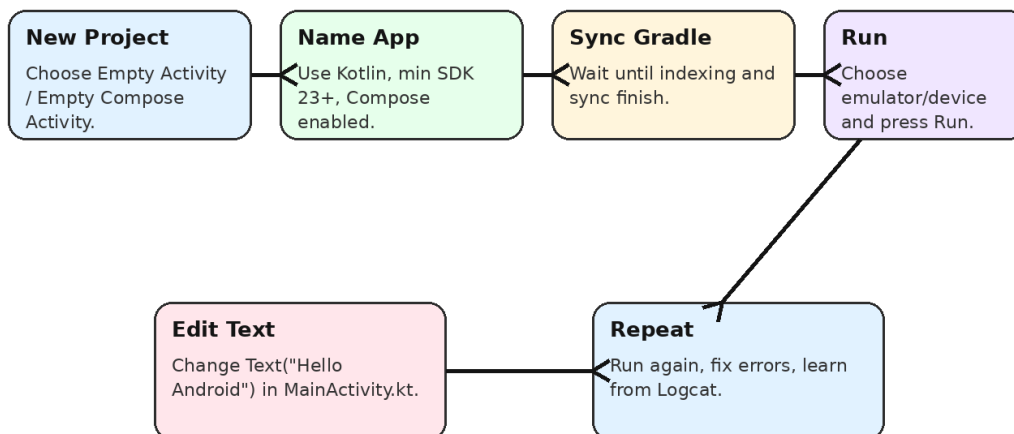
Using a real phone

- On your Android phone, enable Developer Options.
- Enable USB debugging.
- Connect with USB cable.
- Accept the RSA debugging prompt on the phone.
- Select the device in Android Studio and press Run.

Always test on a real device before releasing because emulators are useful but not identical to real phones.

8. Hello World Step-by-Step

Hello World Build Steps



First goal: make one tiny change, run it, and understand where the change appears.

Android Studio Guide by Deekay

Option A: Create it yourself

- Open Android Studio.
- Click New Project.
- Choose Empty Activity or Empty Compose Activity.

- Name the app HelloWorldDeekay.
- Use Kotlin.
- Finish and wait for Gradle Sync.
- Press Run.
- Open MainActivity.kt and change the text.
- Run again.

Option B: Use the included project

This zip includes a folder named HelloWorldDeekay. Open that folder in Android Studio. It contains a basic Compose screen with text and a button counter.

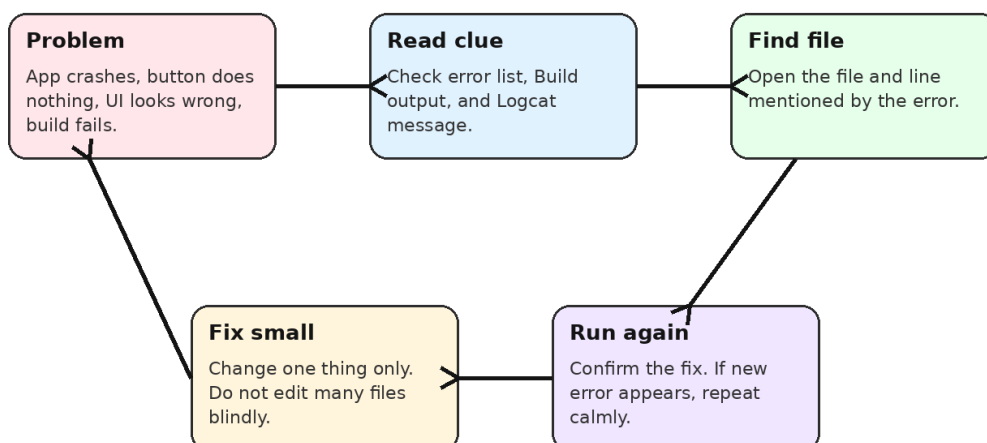
What the included project teaches

- MainActivity extends ComponentActivity.
- onCreate runs when the screen is created.
- setContent starts Compose UI.
- HelloDeekayScreen draws the actual UI.
- remember + mutableStateOf stores click count.
- Button onClick changes state.

9. Debugging Beginner Errors

Debugging is not guessing. Debugging is reading clues. Android Studio gives clues in Build Output, Logcat, and the editor. When something fails, copy the exact error line and find the file and line number mentioned.

Debugging Cycle



Most beginner progress comes from learning to read errors instead of fearing them.

Android Studio Guide by Deekay

Common beginner errors and meaning

Error	Meaning	First thing to try
Gradle Sync Failed	Build setup or internet/dependency issue	Read first red line, check internet, let Studio update plugins
Unresolved reference	Kotlin cannot find a class/function/import	Check spelling and missing import/dependency
Manifest merger failed	Manifest conflict or invalid declaration	Open AndroidManifest.xml and read error details
App crashes on launch	Runtime error after install	Open Logcat and find FATAL EXCEPTION
Emulator slow	Acceleration or low resources issue	Check virtualization, RAM, and use a lighter device image

Beginner rule: fix one error at a time. Do not change ten files because one build failed.

10. Permissions and Privacy Basics

Android permissions protect sensitive data and device features. Some permissions are granted automatically at install. Runtime permissions require asking the user while the app is running. Before requesting a permission, ask whether your app truly needs it.

Beginner permission workflow

- Decide if the feature truly needs private data or restricted action.
- Declare the permission in AndroidManifest.xml if needed.
- Explain the feature in your UI before asking.
- Request runtime permission at the point the user uses that feature.
- Handle deny gracefully. Do not crash or trap the user.
- Remove permissions you no longer need.

```
<!-- Example only: camera permission -->  
<uses-permission android:name="android.permission.CAMERA" />
```

Do not add random permissions. Too many permissions reduce trust and can cause Play Store review problems.

11. App Architecture Basics

Architecture means how you organize code so the app does not become messy. Android's recommended beginner mental model is at least two layers: UI layer and data layer. Bigger apps can add a domain layer between them.

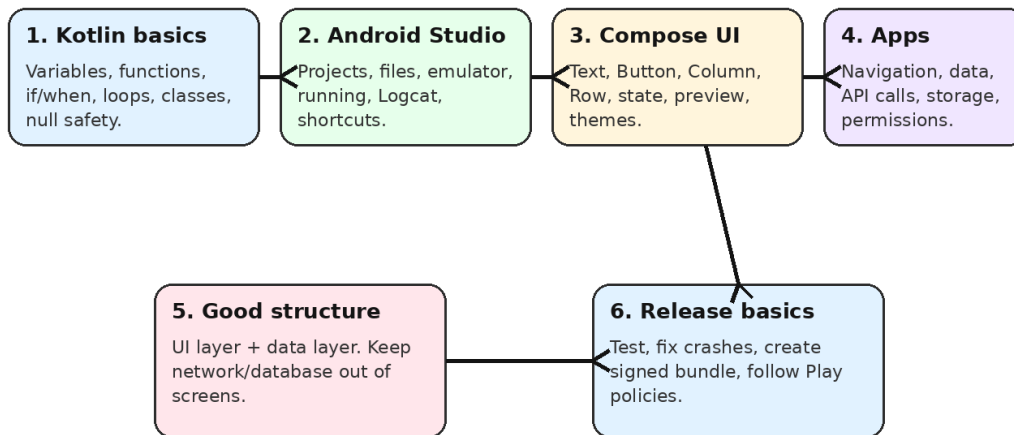
Simple layer meaning

- UI layer: screens, Compose UI, state shown to the user.
- Data layer: repositories, API calls, database, app preferences, business logic.
- Domain layer: optional layer for reusable business rules when the app grows.

For beginner projects, keep screens simple. Do not put all network/database/login/payment logic directly inside a composable function. As your app grows, move business logic into separate classes.

12. 30-Day Learning Path

Beginner Learning Path



Do not rush Java/XML first. For a fresh beginner, Kotlin + Jetpack Compose is the modern route.

Android Studio Guide by Deekay

Days 1-3: Android Studio

Learn Project window, Run button, Build output, Logcat, Device Manager, SDK Manager.

Days 4-7: Kotlin

Learn variables, functions, if/when, loops, lists, classes, null safety.

Days 8-12: Compose

Learn Text, Button, Column, Row, Modifier, state, preview, themes.

Days 13-16: App flow

Learn multiple screens, simple form validation, click actions.

Days 17-20: Data

Learn simple storage, repositories, and basics of API calls.

Days 21-24: Debugging

Learn Logcat, breakpoints, build errors, real-device testing.

Days 25-27: Permissions

Learn manifest permissions and runtime permission requests.

Days 28-30: Mini project

Build one small app: notes, calculator, VPN server-list mockup, or updates app.

13. Best Official Tutorials to Follow

Use official Android learning material first because Android Studio, Gradle, Compose, and APIs change often. Random old tutorials can be useful, but they may use outdated Java/XML patterns or old Gradle versions.

- Android Basics with Compose: best beginner course for no programming experience.
- Create your first Android app codelab: practical first Android Studio project.
- Build a simple app with text composables: learn Compose Text and layout basics.
- Run your first app on the Android Emulator: learn virtual device setup.
- Jetpack Compose tutorial: learn composables, previews, and modern UI ideas.
- Kotlin basic syntax: quick reference for Kotlin language basics.
- Guide to app architecture: learn UI layer and data layer thinking.
- Debug your app: learn breakpoints, variables, and runtime inspection.

14. Sources Summarized

The guide was written in original beginner-friendly wording and summarized from these official/current sources:

- Android Studio overview: <https://developer.android.com/studio/intro>
- Android Studio download/tools: <https://developer.android.com/studio>
- Android Basics with Compose: <https://developer.android.com/kotlin/androidbasics>
- Create your first Android app: <https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app>
- Build simple app with text composables: <https://developer.android.com/codelabs/basic-android-kotlin-compose-text-composables>
- Jetpack Compose tutorial: <https://developer.android.com/develop/ui/compose/tutorial>
- Compose BOM: <https://developer.android.com/develop/ui/compose/bom>
- Run apps on emulator: <https://developer.android.com/studio/run/emulator>
- Run apps on hardware device: <https://developer.android.com/studio/run/device>
- Build and run app: <https://developer.android.com/studio/run>
- Configure build/Gradle: <https://developer.android.com/build>
- Debug your app: <https://developer.android.com/studio/debug>
- Android permissions overview: <https://developer.android.com/guide/topics/permissions/overview>
- Request runtime permissions: <https://developer.android.com/training/permissions/requesting>
- Guide to app architecture: <https://developer.android.com/topic/architecture>
- Kotlin basic syntax: <https://kotlinlang.org/docs/basic-syntax.html>

End of guide. Keep building small projects. Every Android developer started with a simple Hello World app.