GOOGLE AGENT DEVELOPMENT KIT · KOTLIN 1.0

Building an Android AI agent with Google ADK

One agent definition, three model backends: fully on-device with LiteRT-LM, cloud with Firebase AI, or a hybrid of both. Work through the order below — the backend choice constrains everything after it.

3
Model backends behind one `LlmAgent` API — switching is a one-line change

1
Agent definition to maintain, whether it runs on a phone or a server

0
Runtime reflection — tool schemas are generated at compile time by KSP

Pick the backend before you write agent code

BACKEND	RUNS	TOOL CALLING	BEST FOR
LiteRT-LM	On-device (Android + JVM)	**Yes**	Offline, private, low-latency tasks
ML Kit (Gemini Nano)	On-device (Android only)	**Not yet**	Plain chat and generation
Firebase AI	Cloud (Gemini)	**Yes**	Multi-step tool use — the real work

The build order

1. **Add the core dependency**
   `com.google.adk:google-adk-kotlin-core` is all most projects need. Add the KSP processor for generated tools and the webserver module only when you need them.

2. **Annotate your tools**
   Ordinary Kotlin functions with `@Tool` and `@Param`. KSP generates the schemas at build time — type-safe, `suspend`-friendly, and a malformed schema becomes a build error rather than a production one.

3. **Declare the agent**
   Name, description, model, instruction, and `generatedTools()`. Put the behaviour you care about in the instruction — including "verify before you answer", which prevents the most common agent failure.

4. **Serve it and look at it**
   Run `AdkDevServer` for the trace UI on port 8080, then switch to `AdkApiServer` for headless deploys. Both bind to loopback — these endpoints are unauthenticated.

5. **Test the agent, not the app**
   "It ran" is not "it worked". Score which tools it called, in what order — same discipline as any server-side agent.

BEFORE YOU SHIP

ML Kit cannot call tools yet. It ships as a `-beta` pre-release and drops `functionCall` / `functionResponse` parts, so it is chat-only. If a feature means the agent books, changes or sends something, ML Kit is the wrong backend today.

On-device is not the same as capable. A phone-sized model is excellent at classification and extraction and weaker at multi-step reasoning. Use it for that frontier and delegate the rest to the cloud agent in the same hierarchy.