Faysal Aziz

Design Patterns with Kotlin

for Android & Containerizations - 2 days

FA-0165 · 2 days

Software Development · DevOps, Cloud & Infrastructure

Introduction

Hands-on mastery of robust software patterns, containerization, and orchestration—taught with real Linux environments and real-world workflows

---

The digital battlefield has evolved. In boardrooms across Silicon Valley and beyond, CTOs no longer ask if they should embrace mobile-first and cloud-native architectures—they ask how fast they can modernize their Android ecosystems and connected backends before competitors leave them in the dust.

Welcome to the epicenter of that transformation.

This isn't another theoretical deep-dive into Android fragments or another surface-level tutorial on containerization. This is where three decades of hard-won expertise meets the cutting edge of modern Android and backend architecture. Where Kotlin's elegant expressiveness dances with Android Jetpack libraries. Where containerized APIs, services, and Firebase-like backends integrate seamlessly into Android apps through Docker and Kubernetes.

You're here because mediocrity isn't an option.

Every lesson unfolds in the trenches of real Linux environments and Android projects—the same platforms powering the mobile applications millions depend on daily. No sanitized toy problems. Just the raw, unforgiving reality of Android deployments and the backend systems they rely on.

Your instructor has witnessed the evolution from monolithic apps to modular Android architectures and distributed backend services. Survived the dot-com crash. Architected systems that scaled from thousands to millions of users. Debugged failures at 3 AM when millions of dollars and mobile experiences hung in the balance.

This is that wisdom, distilled.

Every pattern explored has been battle-tested in production Android apps. Every technique demonstrated has solved real business problems. Every line of code written together will echo the rhythms of industry-leading practices that separate the professional Android engineer from the amateur.

Step into the arena where Android development meets modern containerized workflows, where your expertise will be forged in the crucible of production-grade systems.

Learning outcomes

By the end of this course, participants will be able to:

- Understand and apply key creational, structural, and behavioral design patterns in **Android Kotlin apps**.
- Apply Android-specific patterns such as Repository, ViewModel, LiveData/Flow, and Dependency Injection.
- Containerize Kotlin-based backend services/APIs that Android apps consume using Docker.
- Deploy and manage containerized services with Kubernetes to support Android clients.
- Leverage Kubernetes-specific orchestration patterns such as Sidecar, Init, Operator, and Probe patterns for Android-backed APIs.
- Debug, iterate, and optimize deployments across Linux servers and integrate seamlessly with Android apps.

Prerequisites

Participants should have:

- Intermediate knowledge of Kotlin programming with Android Studio.
- Familiarity with Android Jetpack components (ViewModel, LiveData, Navigation, etc.).
- Basic understanding of Linux command line and server operations.
- Basic understanding of Docker and Kubernetes concepts (containers, pods, services).
- Access to Linux servers (physical or virtual) to practice Docker/Kubernetes hands-on.
- Gradle-based Android project setup available.

Lab Requirements:

- Android Studio installed and configured.
- VS Code with the SSH plugin for containerization exercises.
- Ports 21 and 22 (outgoing for remote connection establishment).
- Terminal / Console.

- Unrestricted internet access (for Android SDK, Gradle, and remote training servers).
- Optionally, students may also use WSL2 containers for backend work.

Training outline

1. 1. Fundamentals of Design Patterns in Kotlin for Android

Kotlin idioms vs. traditional patterns

- Comparison of Kotlin's functional features with classical OOP design patterns

Creational Patterns in Android:

- Singleton (object declarations in Android), Factory, Builder, Prototype

Structural Patterns:

- Adapter (RecyclerView Adapter), Decorator (UI theming), Proxy, Facade (Service/Repository layers), Composite

Behavioral Patterns in Android:

- Strategy (UI rendering strategies), Observer (LiveData/Flow), State (UI state handling), Template Method, Command (action encapsulation)

Concurrency Patterns:

- Coroutines, Flow, structured concurrency for async tasks and background threading

2. 2. Android Architecture Patterns

- MVVM with Jetpack ViewModel + LiveData/StateFlow
- Repository pattern for data abstraction (Room, Retrofit, Firebase)
- Dependency Injection with Hilt/Dagger in Android
- Clean Architecture layering for large Android apps
- Best practices for modularization in multi-module Android projects

3. 3. Containerization with Docker

- Docker fundamentals: images, containers, layers
- Building images efficiently with multi-stage Dockerfiles
- Dockerizing Kotlin backend services (Spring Boot in Kotlin, Ktor APIs) to support Android apps
- Best practices: slim base images, build profiles, tagging, layering

4. 4. Kubernetes Overview & Pattern Integration

- Kubernetes core concepts: Pods, Deployments, Services, ConfigMaps, Secrets

  Essential container patterns in Kubernetes for Android-backed APIs:

  - **Sidecar** – logging, monitoring, helper processes
  - **Init Container** – pre-boot DB migrations for mobile backends
  - **Adapter & Ambassador** – API gateway/protocol translation
  - **ConfigMap / Secret Pattern** – secure management of API keys for Android apps
  - **Work Queue, Health Probe, Predictable Demands** (scalable APIs for mobile clients)

  Higher-level patterns:

  - **Controller / Operator Pattern** – custom controllers managing backend services

5. 5. Real-World Application: From Android App to Backend Cluster

- Set a baseline: simple Android app consuming a Kotlin microservice
- Containerize and deploy that microservice with Docker and Kubernetes
- Use Init containers for seeding test data, Sidecars for monitoring/logging
- Manage configs and secrets for Android apps (API keys, OAuth tokens) via Kubernetes
- Set up readiness/liveness probes for backend health
- Scale services, simulate backend failure/recovery while Android client adapts

6. 6. Advanced Pattern: Kotlin Operators in Kubernetes

- Introduction to the Operator pattern in Kubernetes
- Designing a Custom Resource Definition (CRD) for Android-specific backends
- Implementing a Controller in Kotlin using fabric8 client
- Deploy and test an operator that manages “MobileBackend” CRD resources

7. 7. End-to-End Integration & Best Practices

  Combine all pieces into a cohesive mobile + backend flow:

  - Android app with Kotlin patterns
  - Containerized backend services
  - Kubernetes patterns and Operator logic
- Debugging techniques on Linux servers: logs, kubectl exec, pod health checks

- CI/CD pipeline overview: build, test, push Docker images, automate backend deployment, update Android client endpoints

---

© 2026 Faysal Aziz · Kuala Lumpur