MDM Platform — Unified Architecture (with JAMF-like Apple path)

# 0) Goals (for Codex/code-gen)

Generate a repo with clear module boundaries and interfaces.  
Support Apple MDM (APNs + ABM/ASM) like JAMF, plus Android and Desktop agents via the same control plane.  
Keep 80%+ of the code reusable across self-hosted/SaaS/hybrid.  
Start with desktop MVP; have Apple MDM slot-in module ready.

# 1) Product Modes (reusability across offerings)

Core (100% shared): policy model, device model, command bus, audit, RBAC, tenancy, API schemas, storage, eventing.  
Interfaces/Adapters (swap per offering): Transport (APNs, FCM, WebSocket/Poll), Persistence (SQLite/Postgres), Auth & Billing, Remote Access.  
  
Structure:  
apps/self\_host, saas, hybrid  
packages/core, adapters, features, ui

# 2) High-Level System (Unification Layer)

Control Plane: Policy Service, Command Service, Enrollment Service, Inventory/Telemetry, Event Bus.  
Platform Adapters: Apple MDM Adapter (APNs + Profiles), Android Adapter, Desktop Agent.  
Unification via normalized Command & Policy Model.

# 3) Apple (JAMF-like) Path — Detailed

Implements /mdm/checkin and /mdm/connect endpoints.  
Push via APNs Provider Token.  
Profile Service signs and tracks Configuration Profiles.  
Command Queue per device (FIFO, retries, ack).  
Certificates: APNs token, optional SCEP/ACME, ABM/ASM for DEP (phase 2).  
Inventory via DeviceInformation, InstalledApplicationList, ProfileList.  
Policies mapped to Configuration Profiles.  
Optional macOS agent complements MDM channel.

# 4) Normalized Models (for code-gen)

Device, Command, and Policy JSON schemas unify Apple/Android/Desktop.  
Example: Device object contains platform, enrollment\_channel, owner, facts, mdm tokens.  
Policy DSL maps rules to platform equivalents (Profiles, registry ops, Android DPM).

# 5) Services & Interfaces (for reuse)

CommandOrchestrator interface to enqueue/ack commands.  
EnrollmentService handles invites and platform check-ins.  
AppleProfileService renders and installs signed configuration profiles.

# 6) Web Console (unified UX)

Devices dashboard, unified policy editor, platform-specific previews.  
Enrollment via QR/profile links.  
Unified event logs and command histories.

# 7) Deployment (three offerings)

Shared 12-factor core; env-driven configs.  
Self-hosted: SQLite, local storage, APNs optional.  
SaaS: Postgres, multi-tenant, APNs per tenant, billing.  
Hybrid: local control + optional cloud relay (WebRTC).

# 8) Implementation Plan (phased)

Phase 0 – Core & Desktop MVP: domain, command bus, policy DSL, agent, console.  
Phase 1 – Apple MDM: APNs, endpoints, plist builder, signed profiles.  
Phase 2 – ABM/ASM + SCEP: DEP zero-touch, extended inventory.  
Phase 3 – Android: Agent polling, later Android Enterprise/FCM.

# 9) Testing Matrix (reusable across offerings)

Contract tests for adapter boundaries, golden files for plist output, end-to-end enrollment to command ack.  
Load tests for queues, tenancy validation for SaaS mode.

# 10) Example API (for code-gen)

POST /api/devices/{id}/commands → enqueue LOCK/WIPE/etc.  
POST /api/policies → define Policy DSL.  
POST /mdm/checkin, /mdm/connect → Apple MDM endpoints.

# 11) Reuse Summary

Core domain, services, and web console shared across all offerings.  
Adapters (APNs, FCM, WS, DB, Billing) are modular.  
Apple MDM plugs into the same command bus and policy renderer.

# 12) Repo Skeleton (ready for scaffolding)

apps/self\_host, saas, hybrid  
packages/core, adapters, features, ui  
agents/desktop (Go)  
infra/docker-compose, k8s, migrations

# 13) What to Build Next

Define Policy DSL + renderers.  
Implement Command Bus + Apple adapter.  
Build Desktop agent.  
Create Web Console.  
Compose self-host mode and document APIs.