ZENTRIX
ENTERPRISE AI GATEWAY

技术白皮书

架构 · 治理 · 安全 · 私有化交付

Zentrix 技术白皮书

企业级 AI 网关的架构设计与工程实践

面向技术决策者、企业架构师与安全负责人，说明 Zentrix 由什么组成、关键机制如何保证可用性与安全性、每项能力如何启用与验证，以及私有化环境下的交付与运维代价。

总体架构 | 模型网关 | 安全与权限 | 数据与可观测 | 可靠性与容量 | 私有化交付 | 开放性

文档类型	适用产品	文档版本	对应基线
技术白皮书	Zentrix Gateway	2026.08	main · 3cbd59bd5d

目录

章节	主要内容
第 1 章 · 概述与产品定位	Zentrix 要解决企业接入大模型的哪四个约束、由哪 31 个服务承担、每个约束的落点在哪里，以及全书统一的能力标注口径
第 2 章 · 总体架构	Zentrix 单机编排下的 31 个服务、四层调用边界、四条请求主链路、入口站点划分与运行时配置代际闸
第 3 章 · 模型网关核心机制	Zentrix 的产品本体——七层判定链、供应商四事实精确解析、三种"切换"语义、三套限流各自的失败姿态、数据面插件链与计量口径
第 4 章 · 安全架构	授权、密钥、租户隔离、网络与宿主边界、内容安全、审计六条可独立验证的控制面，以及 Zentrix 形态下的默认姿态、启用方式与刻意 fail-open 的位置
第 5 章 · 数据架构与可观测性	Zentrix 的数据存在哪、谁能连、留多久、怎么演进，以及它向外输出哪些可观测信号、你如何把这些信号接进自己的监控体系
第 6 章 · 可靠性、容量与性能	它坏了会怎样、多久能修好、一台机器能撑多少，以及最后一个问题为什么本书给方法而不给数字
第 7 章 · 私有化交付与运维	Zentrix 的单包交付边界与 31 个编排单元、五道介质校验、把部署配置编译成一代的事务模型，以及整包升级与配置回滚各自的真实边界
第 8 章 · 开放性与集成	Zentrix 的对外协议兼容面、管理规格与审计导出、MCP 接入与分发、网关插件与路由策略两个扩展位，以及与既有身份、监控体系的对接方式
第 9 章 · 技术选型说明与取舍	逐条交代十个关键技术选择的备选方案、放弃了什么以及代价是什么
附录 A · 组件、端口与能力一览	Zentrix 交付包的 31 个服务、端口暴露面、网络分区默认值，以及可启用能力与开关的集中清单
附录 B · 术语表与命名对照	本书术语的统一写法、代码侧对应名、一句话定义与禁止写法

关于本文档

本文档说明 Zentrix 企业级 AI 网关的技术组成、实现机制与工程约束。

一条编写原则

本文的全部技术描述以产品代码基线 main `3cbd59bd5d`（2026 年 8 月 17 日）为准。文中出现的端口、默认值、上限、开关名与执行顺序，均取自该基线的源码、编排文件与数据库迁移，而不是产品宣传材料或界面文案。

这条原则带来的直接后果是：本文对每一项能力都写到可以被现场核验的粒度——它由哪个开关启用、启用后影响哪一层判定、适用范围到哪里、每个技术选择付出了什么代价。我们认为对做技术尽调的读者而言，这些信息比形容词更有价值；能被验证的描述，才是可以写进合同的描述。

读者对象

读者	关注的章节
技术决策者、选型负责人	第 1 章产品定位、第 9 章技术选型与取舍
企业架构师	第 2 章总体架构、第 3 章模型网关、第 5 章数据架构
安全与合规负责人	第 4 章安全架构、附录 A 组件与端口清单
平台运维负责人	第 6 章可靠性与容量、第 7 章私有化交付与运维
应用集成方	第 3 章模型网关、第 8 章开放性与集成

产品口径

Zentrix 是可独立交付的企业级 AI 网关，负责模型接入与路由、流量保护、调用治理与凭据分发、内容护栏、调用可观测与计量。它以私有化离线包形式交付，共 31 个服务，自带完整的控制面、能力治理面与全套有状态存储，不依赖任何外部托管服务。

> **说明**
>
> Zentrix 与 Zenova 的关系：同一套安装器与配置体系之上，Zentrix 加装智能体运行时（婉儿）闭包即构成 **Zenova** AI 智能办公私有化平台。本文以 Zentrix 为主体，Zenova 的扩展路径只在第 1 章末尾交代，智能体运行时的机制细节不在本文范围。

能力标注

本文对每项能力都写明两件事：**它怎么启用**，以及**它的适用范围到哪里**。统一采用两种标注：

标注	含义
默认启用	随标准交付包部署即生效，无需额外配置。
可配置启用	能力随包就位，通过明确的开关启用；本文写明开关名、启用后的影响与需要在交付时做的配置动作。

这两种标注覆盖本文提到的每一项能力：读到"可配置启用"时，你能同时拿到开关名与启用后的效果，因而可以把它直接写进交付验收清单。代码默认值与出货交付包默认值不一致时，本文两个都写——只写其中一个会误导容量规划与安全评估。

阅读约定

高危
不可逆操作或影响面覆盖整套部署的约束，选型与实施阶段都需纳入决策。

注意
常见误解，或与直觉相反的实际行为。

说明
补充背景、设计取舍或适用范围。

能力状态
标明该能力是默认启用，还是通过指定开关配置启用。

本文正文不出现可直接执行的命令与操作步骤，它们属于《Zentrix 安装与配置指导》；界面路径与角色操作旅程属于用户手册。需要指代某个动作时，本文用能力名称描述。

第 1 章 · 概述与产品定位

1.1 问题域：卡住项目的四个硬约束

企业把大模型接进生产系统时，最先卡住项目的通常不是模型效果，而是四个与"这次调用返回了什么"无关的约束。它们的共同点是：产生位置都在一次模型调用之外。

约束	它要回答的问题	产生位置
数据不出域	提示词、调用日志、凭据与中间产物落在谁的磁盘上	部署拓扑
权限可审计到人	这次操作代表哪个自然人、他当时具备哪些角色、谁能事后证明	身份与留痕体系
成本可归因到组织单元	这笔支出算给哪个租户、哪个工作空间、哪个调用方	记账口径
故障可定位到单次调用	哪一次请求、走了哪条路由、被哪一层以什么原因拒绝	判定链的留痕粒度

表 1–1　四个硬约束及其产生位置

这四条不是"更高的安全要求"，而是四类必须有人承接的工程职责。谁都不承接时，它们不会报错，只会在事后审计、成本对账或故障复盘的时候同时失效。

1.2 为什么公有云 API 加一层自研胶水补不上这四条

公有云模型 API 的语义边界只到一次调用：请求进去，结果出来。上表右列的四个产生位置全部不在这个边界内，因此通常由调用方自己写一层胶水补齐——在应用里塞租户上下文、在网关前加一段日志、在下游拼一张成本报表。

胶水能做到相似的效果，代价是把这四条约束变成了应用层的自律。某个新接入的服务忘了带租户上下文，某次上线漏了一段脱敏，某个后台任务直接绕过了统一入口——系统不会拒绝它们，它会照常返回 200。等到需要用这些数据回答"这次操作是谁做的""这个月的账怎么拆"时，缺口已经积累了几个月，而且无法追溯补齐。

逐条看，四个约束在"API 加胶水"这条路上各自缺一块地基：

- **数据不出域**缺的是部署单元。只要模型推理与调用日志落在别人的机房，任何应用层措施都只能减少泄露面，不能改变数据的物理归属。
- **权限可审计到人**缺的是权限权威。胶水里的权限判断分散在各个服务的条件语句中，没有一个可以被单独审查的地方能回答"这条授权是从哪一条关系推导出来的"。
- **成本可归因**缺的是一手记录。事后从应用日志聚合出来的用量是二手推算，与上游账单对不齐时无法判断谁错。
- **故障可定位**缺的是判定链。请求要么成功要么失败，中间被哪一层拦下、依据哪条策略，胶水层通常没有记录，因为拦截动作本身就散落在各处。

Zentrix 的做法是把这些约束下沉到部署单元与请求期判定里，让"前提不满足"表现为拒绝服务，而不是记一条告警继续跑。四个可验证的例子：

1. 护栏服务地址未配置时，控制面与受治理调用执行面拒绝启动，而不是跳过内容检查。
2. 内存态的淘汰策略被改成会淘汰键的策略时，限流中间件在请求期判定并直接返回 503，因为一个会被淘汰的计数器等于没有限流。
3. 运行时配置未编译成当前代际时，编排文件顶部的代际标签取不到值，一个容器都创建不出来。

4. 每次启动前有一个一次性的代际校验闸：它在任何服务碰数据库之前，把镜像内嵌的配置登记表摘要与本地投影摘要做比对，不一致就让整次启动失败。这道闸不接任何网络、只读挂载安装目录，三个交付包都带它。

第 4 条针对的失败很具体：用旧配置投影喂新二进制，现象是某个服务反复重启并报一个指不到根因的缺参数错误。把它前移成启动前的一次摘要比对，等于用一次确定的失败换掉一类难以归因的崩溃循环。

> **说明**
> 本章只界定边界。四条约束分别由哪一层机制承接——授权与留痕、密钥与租户隔离、计量与用量投影、判定链与拒绝原因——在后续章节展开。

1.3 Zentrix 是什么：一个 31 个服务的完整交付单元

Zentrix 是可独立交付的企业级 AI 网关，能力范围是五件事：模型接入与路由、流量保护、调用治理与凭据分发、内容护栏、调用可观测与计量。

它在代码侧对应一个独立的发行形态，不是文案划分：该形态的产品名字段取值为 Zentrix，显示名为 Zentrix Gateway；它使用一份替换版的入口配置文件，并对关系主库指定了一组该形态专属的启动参数（共享缓冲 256 MiB、有效缓存估计 512 MiB、单次排序工作内存 8 MiB、最大连接数 500、口令加密方式 scram-sha-256、预写日志级别 replica）。容器引擎与容器编排等操作系统级前置依赖由目标机器提供，这一点与另外两个包一致。

> **常见误解**
> Zentrix **不是"一个网关进程"**。它的服务白名单里有 **31 个服务**，包含完整的控制面、能力治理面、护栏与全套有状态存储。把它理解成一个反向代理加计费，会在容量规划、故障域划分与运维排班上全部估错。

31 个服务分九组：南北向入口与网关证书热重载（2）；模型网关数据面及其库初始化任务（2）；控制面、平台库迁移器与按需拉起的管理动作工具容器（3）；能力平面、受治理调用执行面与运行时数据面前置（3）；护栏服务及其内置规则种子与模型工件装载任务（3）；计量、告警评估、维护清扫三个独立执行进程与通知适配器（4）；密钥服务、密钥库迁移器与两条工作负载令牌轮换（4）；配置代际校验闸（1）；关系主库与连接池、授权服务及其独立库与迁移、内存态、对象存储与其初始化、列式库共九个有状态组件（9）。

三个独立执行进程的职责分工，包配置里逐条写明了：用量投影由计量进程执行，控制面内的同名开关保持关闭以避免重复投影；告警规则的评估循环由告警进程承担，控制面只保留告警的管理接口；会话过期清理与审计留存清理由维护进程承担。换句话说，这三个进程不是可选装饰，是这些能力的唯一执行者——交付包按"哪条链路由谁执行"组织，而不是按目录组织。

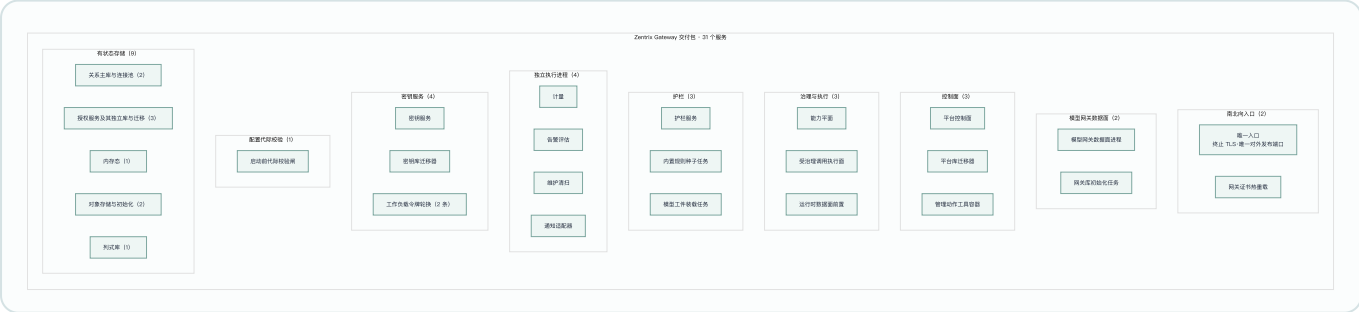


图 1–1　Zentrix 交付包的九组服务

这张图回答一个问题：选 Zentrix 之后，机器上会跑起哪些东西。服务清单由发布配置的服务白名单驱动渲染，白名单之外的服务连同指向它的启动依赖一并不进入编排文件，因此拿到手的是一份自洽的编排，而不是"装了但不启动"。图中不含请求流向，链路见图 1–2。

1.4 三个易被混称为"网关"的进程与本书的命名约定

这套系统里有三个进程的代码名都带 router 或 gateway，职责却完全不同，混称为"网关"是最常见的误读来源。本书对它们各用一个固定中文名，全书不换写法；代码侧进程名只在需要精确指代进程时出现，禁止写法集中登记在附录 B。

本书统一写法	代码侧进程名	在 Zentrix 形态下承担什么
模型网关数据面	`zstack-router`	模型调用的实际承载者：客户端来源归一、域名与 HTTP 策略、凭据认证、准入判定、统一路由策略、插件链、供应商适配与协议变换、上游调用与逐次用量记账
受治理调用执行面	`lp-agent-gateway`	受治理的 MCP 与连接器调用的单次执行者。调用方经控制面取得会话令牌与凭据租约后，由它在已验证租约上自行解密凭据并派发到上游系统；凭据不下发给调用方。它还持有代操作用户请求头的私钥签名文件
运行时数据面前置	`lp-gateway`	持有控制面签名下发的策略快照并维护本地台账；另有三组内部路由（控制台访问准入、模型出口代理、MCP 清单查询）均以智能体身份为前提

表 1–2　三个易混称进程的统一写法与职责

前两个在 Zentrix 形态下都有对外链路。受治理调用执行面尤其值得点名：Zentrix 的入口配置显式为受治理 MCP 的流式端点保留了一条独立路由，第三方 MCP 客户端与企业内应用可以经这条链路调用被治理的外部系统，全程不需要智能体容器参与。

> **说明**
>
> 运行时数据面前置（`lp-gateway`）随包交付并常驻运行：它与控制面之间维持签名策略快照的续期与落盘，并把快照写进本地台账，重启时先从磁盘装载未过期快照。它那三组路由以智能体身份为前提，在加装智能体运行时（见 1.8）后投入使用。本书在服务清单里如实列出它的这一职责范围。

1.5 四个约束在 Zentrix 里的落点

数据不出域：默认交付形态是单机容器编排栈，31 个服务与全部有状态存储都跑在企业自己的一台机器上。唯一在宿主全部网络接口发布端口的服务是南北向入口（80 与 443），其余服务的宿主端口一律绑回环。上游模型端点由企业自己配置，网关不预置可直接生效的上游——供应商目录里的默认基址只是管理台的便利值，运行时不以它为源权威；上游 TLS 信任操作系统的证书库，因此企业自建、由内网私有 CA 签名的模型端点可以直接接入。

权限可审计到人：权限权威是一个独立的关系型授权服务，模型定义在仓库内的单文件 schema 中，由控制面在启动时做差分迁移。租户侧实装等保 2.0 三员分立（配置管理员、安全管理员、审计员互斥），平台侧另有一套角色，且平台四角色不隐含租户管理权。判定强一致，进程内只缓存肯定结果且生存期极短；授权服务不可达时一律拒绝而不是放行。留痕侧：审计表有触发器强制追加写，另有 16 分片的哈希链且没有逃生口，审计事实默认保留 730 天。租户数据隔离另有一道：81 张表强制行级安全，其中 13 张故意不设策略等价全拒；谓词基于事务级会话变量，未设置即返回 0 行而不是返回全部。

成本可归因：一手记录是模型网关数据面自己的用量日志，逐次调用落库。计量执行进程按固定周期（15 秒，且下限被强制为 15 秒）用游标从数据面拉取并投影到关系库，产品侧的用量与成本口径读投影表。归因维度是租户、工作空间与调用方三层；调用方密钥自带有效期、吊销时间与吊销原因、绑定的模型服务与计费绑定、声明的客户端类型，因此"停掉某个应用的开支"是吊销一把密钥，而不是改一段代码。

故障可定位到单次调用：每一层判定都有自己的拒绝原因标识，而不是统一落成 500。凭据校验区分凭据无效、主体被禁用、凭据过期、来源 IP 不在白名单；模型白名单在任何路由工作之前评估，拒绝时返回 403 并带独立的策略决策标识；域名层的明文请求策略有放行、跳转 HTTPS、拒绝三种模式，非可信代理来的明文请求在非放行模式下直接拒并单独计数。凭据校验每次请求都查库、不设认证缓存，因此"刚吊销的密钥还能用一会儿"这种排查噪声不存在。

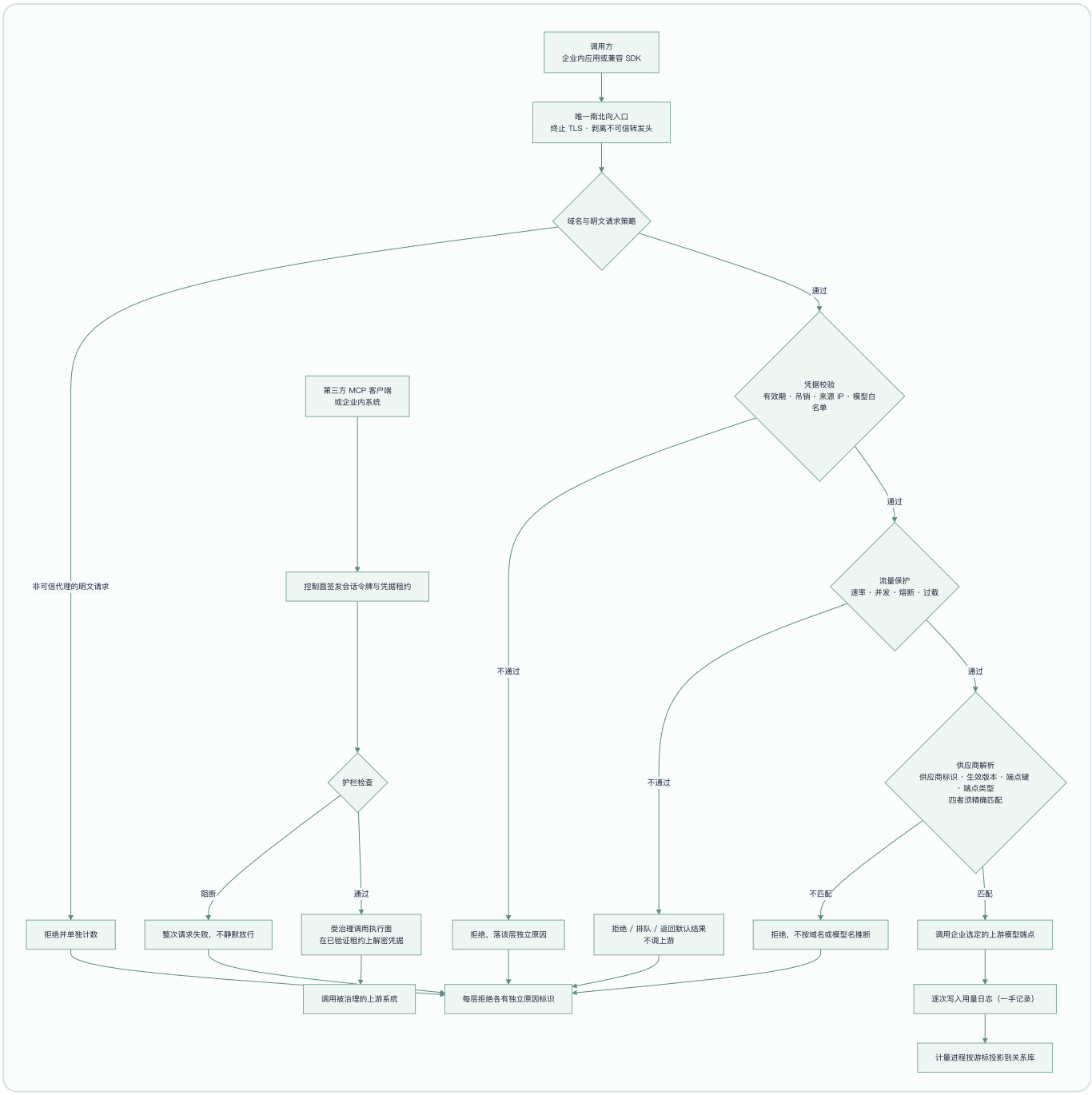


图 1–2　Zentrix 的两条对外链路与它们的拒绝分支

这张图回答两个问题：一次模型调用会经过哪些可判定的关口；受治理能力调用走的是另一条链路。注意护栏不在模型调用链上，它挂在受治理能力与对话路径上——把它画进模型链路是常见的画错法。

1.6 交付形态对比：Zentrix 与另外三个包

同一套安装器与配置体系渲染出四个交付包。下表以 Zentrix 为主列。

项	Zentrix Gateway	POC	Full	一体机
定位	可独立交付的企业级 AI 网关	售前验证与快速试点	智能办公平台正式私有化主线	介质封装形态
服务数	**31**	34	52	继承 POC
镜像条目数	23	37	56（高可用闭包 64）	继承 POC
入口配置	替换版（网关形态专用）	基线版	基线版	同 POC
关系主库启动参数	本形态专属覆盖	基线	基线	同 POC
护栏能力档位	安全词库 + 规则式个人信息识别	同 Zentrix	另含语义安全模型（必需）与实体识别（可选）	同 POC
编排档位	护栏、护栏模型	护栏、护栏模型	可观测栈、护栏、护栏模型	同 POC
智能体运行时（婉儿）	加装即成为 Zenova（见 1.8）	含（POC 专用镜像）	含（完整镜像）	同 POC

表 1–3　四个交付包的构成差异

四个包由同一份编排基线按服务白名单渲染：完整形态就是基线本身（52 个服务、22 个命名卷、3 张网络，全部跑在一台机器的容器编排栈上），POC 渲染出 34 个服务，Zentrix 渲染出 31 个，一体机继承 POC 已验证的镜像闭包、差别在介质封装。完整形态与 Zentrix 另外声明了一套面向 Kubernetes 的高可用部署模式，其成熟度按现状标注（chart 版本 0.1.0，离线升级经镜像仓库策略交付）；单机容器编排是默认与主线。

> **形态差异**
> Zentrix 形态的护栏装配安全词库与规则式个人信息识别两类引擎；语义安全模型与实体识别由完整形态的能力档位装配。评估护栏能力时请按本形态的档位口径读（见第 4 章表 4–3）。

1.7 能力标注的两种口径

本书对每一项能力只写当前基线可验证的启用方式，两种口径互斥。

标注	含义	Zentrix 形态下的示例
默认启用	标准安装完成即生效，无需额外配置	唯一南北向入口对外发布 80 与 443、其余服务宿主端口一律绑回环；关系型授权判定；护栏作为控制面与受治理调用执行面的硬依赖，未配置服务地址即拒绝启动；启动前的配置代际校验闸；内存态淘汰策略必须为不淘汰，否则限流中间件在请求期返回 503；用量投影按 15 秒周期运行

标注	含义	Zentrix 形态下的示例
可配置启用	能力随包就位，通过明确的开关启用；必须写出开关名与启用后的影响	出网 SSRF 守卫（平台侧开关 `ssrf_guard_enabled` 、模型网关数据面侧开关 `ssrf_settings.enabled` ，两者都是设置项而非环境变量，按客户的出网策略启用）；代操作用户请求头的签名封装（`LP_ACTING_USER_HEADER_JOSE_ENABLED` ，控制面与受治理调用执行面两端成对启用）；22 条内置个人信息识别规则（按客户的合规要求逐条启用）；异地备份拷贝定时器（`LP_BACKUP_RESTORE_OFFHOST_SCHEDULE_ENABLED` ，随包安装、默认禁用，且需计划、平台密钥与计划令牌三者齐备才会调度）

表 1–4　能力标注的两种口径与本形态示例

> **能力状态**
>
> "可配置启用"在本产品里落在两个层面，本书分开写：**编排档位**——服务随包交付，启用对应档位即投入运行；**应用开关**——服务已在运行，改一项配置即生效。两者的启用代价不同：前者改编排并重新拉起相关服务，后者走一次配置事务（见第 7 章）。本书对每一项都写明它属于哪一层。

还有两条贯穿全书的写法约定。其一，同一项配置在代码默认值与出货编排默认值不一致时，本书两个都写，并标明哪一个在标准交付中实际生效——只写一个会直接误导容量规划与安全评估。其二，本书不写路线图、不写"将来会有"，只描述当前基线上可以被现场验证的行为。

1.8 可扩展路径：加装婉儿运行时后成为 Zenova

Zentrix 与 Zenova 共用同一套安装器、同一套配置编译与代际切换体系、同一份编排基线。两者的差别是一个字段加一批服务：Zenova 的交付包声明了智能体运行时镜像，并在同一份基线上多渲染出一批服务（智能体运行时闭包、备份工具轨道、组件更新服务等）。婉儿（Waner）是 Zenova 的智能体运行时模块，以运行时镜像的形式跑在每一个智能体容器内部；容器的创建、销毁与回收由控制面掌管，婉儿不是编排者。加装它之后，产品从"企业级 AI 网关"变为"AI 智能办公私有化平台"，而 Zentrix 已经具备的授权、租户隔离、护栏、计量与配置事务这套治理底座保持不变。

婉儿的机制细节不在本书范围内。

> **注意**
>
> 这条扩展路径是**换交付包**，不是打开一个开关。安装器对跨形态的原地升级做硬阻断：先按包文件名预警，解包后再用包内清单做权威校验并拒绝。规划从 Zentrix 走向 Zenova 时，必须按"换包部署 + 数据迁移"来排期与验收，不能按"升级"估算窗口。

1.9 这对你意味着什么

如果你是选型负责人，本章给你的动作不是对功能清单打勾，而是先确认三件事：你要解决的问题是不是落在模型接入与治理这一段（需要智能体运行时的场景走 1.8 的扩展路径）；31 个服务（其中九个有状态）是你要长期运维的真实规模，磁盘、备份与升级窗口按这个数算，不按"一个网关进程"算；四个硬约束里哪几条是验收项——把它们对到 1.5 节的落点，要求在你自己的环境里演示"前提不满足时它确实拒绝"，而不是"满足时它能跑通"。

第 2 章 · 总体架构

第 1 章界定了产品边界。本章回答架构师在评审会上最常追问的那一个问题：**这套东西装进我的网络里，会长成什么样。** 下文按"一句话架构 → 分层与调用边界 → 组件职责 → 四条请求主链路 → 入口面 → 网络分区 → 一致性闸"的顺序展开：每一跳标出协议、端口与认证方式，每一处默认值都取自当前基线的编排文件与代码，能力一律按**默认启用 / 可配置启用（写明开关名）**两种口径标注。

2.1 一句话架构与交付形态

Zentrix 的默认交付形态是**一台机器上的单机容器编排栈：31 个编排服务、14 个命名卷**，全部跑在同一台宿主机上。这 31 个服务不是"网关进程加一个数据库"——它包含控制面与管理后台、能力平面与受治理调用执行面、护栏、告警与计量、密钥服务全家、模型网关数据面，以及关系主库、连接池、内存态、授权服务、对象存储、列式库这一整套有状态存储。

这 31 个服务由发布配置的服务白名单驱动渲染：白名单之外的服务连同指向它的启动依赖一并不进入编排文件，因此拿到手的是一份自洽的编排文件，而不是"装了但不启动"。同一份编排基线因此可以按交付形态产出不同规模的部署，而每一份都能被逐个服务地枚举与核对。

Zentrix 的关系主库另有一组本形态专属的启动参数：共享缓冲区 256 MiB、有效缓存估计 512 MiB、单次排序工作内存 8 MiB、最大连接数 500、口令加密算法固定为 SCRAM-SHA-256、预写日志级别为副本级。这组参数随交付包一起渲染进编排文件，不需要运维手工设置。

> **说明**
> "31 个服务"不等于"31 个常驻进程"。其中包含一次性初始化与迁移任务（跑完即退出）、只在特定运维动作时才被拉起的工具容器，以及护栏模型资产的一次性装载任务。真正常驻的服务数量小于这个数字。

除单机编排外，Zentrix 在发布包配置中另有一条 Helm 交付路径。**本书把单机编排作为默认与主线口径描述；Helm 路径的成熟度按现状单独标注，见第 7 章。** 本章其余部分如无特别说明，均指单机编排形态。

2.2 分层模型与调用边界

Zentrix 按四层组织。分层的意义不在于画得好看，而在于每一层都有一条可以被验证的"不能调谁"。

层	承担什么	硬边界
入口层	TLS 终止、静态资源、请求转发、模型网关对外站点	唯一在宿主全部网络接口上发布端口的层
控制面	平台 API、会话与身份、管理后台后端、配置与运行状态	不直接暴露给外部；持有宿主容器运行时套接字的唯一服务
治理与执行面	能力投影与绑定、受治理调用的执行与凭据解密、护栏、告警、计量、维护、模型网关数据面	能力平面与受治理调用执行面不回调控制面做鉴权
数据面	关系主库、授权服务、内存态、对象存储、列式库、连接池、密钥权威	全部不在宿主全部网络接口上发布端口

表 2-1　四层模型与各层的硬边界

可观测信号由各服务自身输出：每个服务在自己的端口上暴露低基数指标端点，结构化业务事件与容器日志按固定格式落盘。把这些信号接进客户既有观测体系的具体方式见第 5 章与第 8 章。

三条边界值得单独说明，因为它们决定了故障面怎么切：

其一，能力平面直读关系库与授权服务，绝不回调控制面做鉴权。 "某个调用主体现在能用哪些能力"这个判定，不会因为控制面繁忙或重启而失效，也不会把控制面变成治理路径上的单点。代价是能力平面必须自己持有数据库与授权服务的连接与凭据。

其二，受治理调用执行面是唯一持有调用者身份签名私钥的服务，该私钥文件不挂载到控制面。外部系统凭据在这一跳解密并派发，不写回调用方，也不从请求体读取。该身份头的双层签名封装由 `LP_ACTING_USER_HEADER_JOSE_ENABLED` 启用，控制面与执行面两侧成对开启；私钥的挂载边界不随该开关变化，始终只在执行面。

其三，模型网关数据面不依赖控制面的健康状态。 它只等控制面"已启动"，不等"已健康"——模型流量的可用性不被管理面绑架，这一条在第 6 章还会展开。

> **注意**
> 护栏对控制面与受治理调用执行面都是硬依赖：未配置护栏服务地址时，这两个服务拒绝启动，而不是绕过内容检查继续跑。这是本产品 fail-closed 姿态的一个典型位置，第 4 章会给出完整清单。

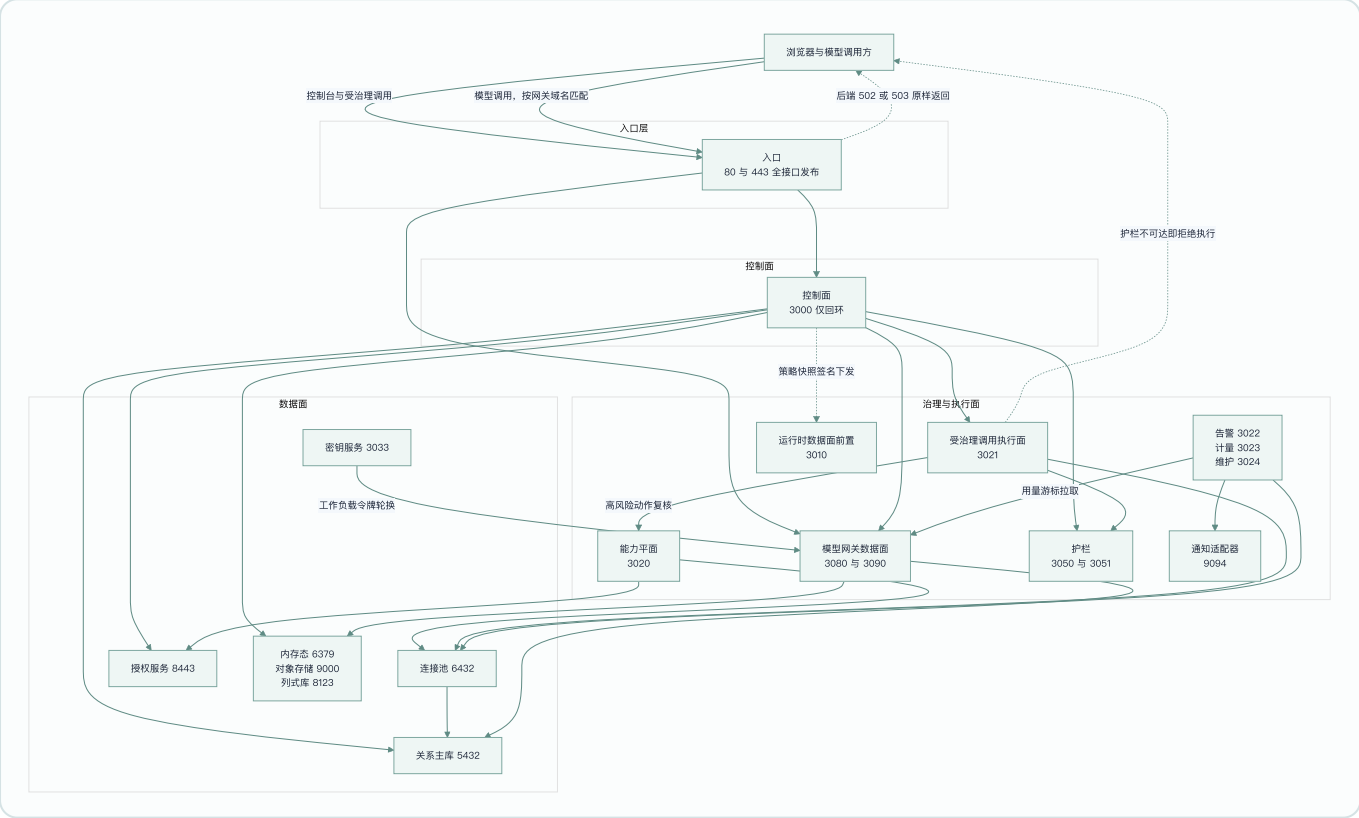


图 2–1　Zentrix 分层组件全景

这张图回答的问题是：一次外部请求从哪一层进入、在哪一层被拦下、哪些组件之间不允许直接互相调用。虚线表示旁路、降级或拒绝路径，实线表示请求主路径。

2.3 组件职责边界

下表列出平台侧服务的职责边界与暴露方式，端口一列为容器内监听端口。

组件	职责边界	端口	宿主暴露
入口	TLS 终止、静态资源、请求转发、模型网关对外站点	80 与 443	全接口发布
控制面	平台 API、会话与身份、管理后台后端、配置与运行状态	3000	仅回环

组件	职责边界	端口	宿主暴露
管理任务容器	平台管理动作的一次性执行体	—	不发布
模型网关数据面	模型调用的准入、路由、上游适配与用量记账	3080 与 3090	不发布
能力平面	有效能力投影、绑定读写、高风险动作复核；直连关系库与授权服务判定	3020	仅回环
受治理调用执行面	受治理 MCP 与连接器的单次调用执行；凭据在本服务内解密	3021	仅回环
运行时数据面前置	持有控制面签名下发的策略快照，控制面不可用时按快照继续判定	3010	仅回环
护栏	内容安全与个人信息保护的统一判定	3050 业务、3051 运维	不发布
告警服务	告警规则评估与事件投递	3022	仅回环
计量服务	用量投影、配额重置、工作空间运营汇总	3023	仅回环
维护服务	短周期清扫，含审计留存清理与日志留存清理	3024	仅回环
通知适配器	把告警事件翻译成各通知渠道的报文格式	9094	不发布
密钥服务	密钥权威与工作负载令牌轮换	3033	不发布

表 2-2　平台侧组件的职责边界与暴露方式

关于"运行时数据面前置"这个名字：该进程随包运行的职责是与控制面之间的签名策略快照续期与落盘，外加自身的就绪与指标端点；它的三组接口——访问前置鉴权、助手模型出口代理、能力服务清单——都以智能体身份为主键，在加装智能体运行时后投入使用。本书据此命名，并在附录 B 登记这一口径。

> **注意 · 容器运行时套接字的挂载边界**
>
> 编排文件把宿主的容器运行时套接字挂进容器，且是可读写挂载。**Zentrix 形态下只有控制面一个容器持有它**，用于运行概览的容器资源统计与留存策略变更时的编排应用。需要提醒的是：**把该套接字以只读方式挂载并不构成权限边界**——只读挂载仍允许通过容器 API 执行创建、执行与停止等写操作。真正的边界是"哪些容器拿得到它"，这条口径与第 4 章一致。

> **注意 · 计量与留存清理由独立进程执行**
>
> 计量与留存清理的执行者是计量服务与维护服务，两者的对应开关在出货编排中为真；控制面侧的同名开关 `LP_SERVER_ZR_USAGE_PROJECTION_ENABLED`、`LP_SERVER_QUOTA_RESET_ENABLED`、`LP_SERVER_AUDIT_RETENTION_PURGE_ENABLED`、`LP_SERVER_LOG_RETENTION_CLEANUP_ENABLED` 保持关闭，以保证同一份工作只有一个执行者。做容量评估与故障定位时，请把这两项工作算在独立进程上。

数据库连接拓扑有一条容易写反的边界：**控制面、迁移器、护栏与模型网关数据面直连关系主库的 5432 端口**（会话级顾问锁需要会话连接），**经连接池 6432 端口的是拆分出来的服务**——能力平面、受治理调用执行面、告警、计量与维护。按"每个调用方经连接池连库"来推算连接容量是错的。

2.4 四条请求主链路

2.4.1 控制台管理链路

跳	协议与端口	认证方式
浏览器 → 入口	HTTPS 443 或 HTTP 80，宿主全接口	传输层终止，无应用认证
入口 → 控制面	HTTP 3000，容器网内	转发头附带跳板共享密钥
控制面 → 授权服务	HTTP 8443，容器网内	Bearer 令牌，由主密钥派生
控制面 → 关系主库	数据库协议 5432，直连不经连接池	SCRAM-SHA-256
控制面 → 护栏 / 对象存储 / 列式库	HTTP 3050 / 9000 / 8123	调用方密钥 / S3 签名 / 口令，均由主密钥派生

表 2-3　控制台管理链路的逐跳协议与认证

用户身份为 JWT，签名密钥同样由主密钥派生。跳板密钥这一跳值得单独说明：控制面对该头做的是**共享密钥的常数时间比较**，不是逐请求签名。比较不通过时，控制面不会拒绝请求，而是**丢弃请求里的客户端 IP 转发头、改用 TCP 对端地址**；比较通过时，它把这些头规范化重写为单一可信值。这条设计的目的是让"伪造来源 IP"在任何情况下都拿不到收益。

失败分支：Zentrix 的替换版入口配置**不做后端错误的维护页回落**，后端返回 502 或 503 时按原状态码返回。交付包另带一份维护态入口配置，作为运维动作显式切换的独立配置文件。

2.4.2 模型调用链路

这是 Zentrix 最重要的一条链路，也是唯一一条外部流量可以不经控制面直达执行面的链路。

跳	协议与端口	认证方式
外部调用方 → 入口	HTTPS 443，按 SNI 匹配 `*.{网关域名}` 站点	入口终止 TLS，证书取自平台写入的只读卷
入口 → 模型网关数据面	HTTP 3080，容器网内	跳板共享密钥头，叠加源地址网段白名单
内部调用方 → 模型网关数据面	HTTP 3080，容器网内 DNS，无宿主端口	控制台身份的认证模式默认为远端身份鉴权，开关 `ZR_AUTH_MODE` 默认 `remote_iam`
模型网关数据面 → 网关库	数据库协议 5432，直连	SCRAM-SHA-256
模型网关数据面 → 上游供应商	HTTPS，按渠道配置的上游地址	渠道凭据；上游 TLS 信任操作系统根证书库

表 2-4　模型调用链路的逐跳协议与认证

调用方凭据分两套，落在同一个监听端口的不同路径上：**网关原生入口**用网关自有的调用方密钥体系，**平台兼容入口**用平台侧按租户与工作空间发放的网关模型密钥。两套都在每次请求时查库校验，**没有认证缓存**。模型白名单在任何路由工作之前评估，越权模型直接拒绝。

来源校验是**双闸**：既校验跳板共享密钥头，又用可信代理网段把源地址限定为入口在容器网上的静态地址（默认 `10.244.1.10/32`）。两道闸缺一不可——容器网桥环境下仅凭源地址无法区分同网服务。模型网关数据面自身的工作负载身份是一枚有效期 120 秒的令牌，由独立的轮换器每 30 秒刷新一次。

编排为三个内部调用方配置了同一个容器网内地址：控制面（对话与助手路径）、计量服务（用量拉取）与运行时数据面前置（模型出口路径，见 2.3）。

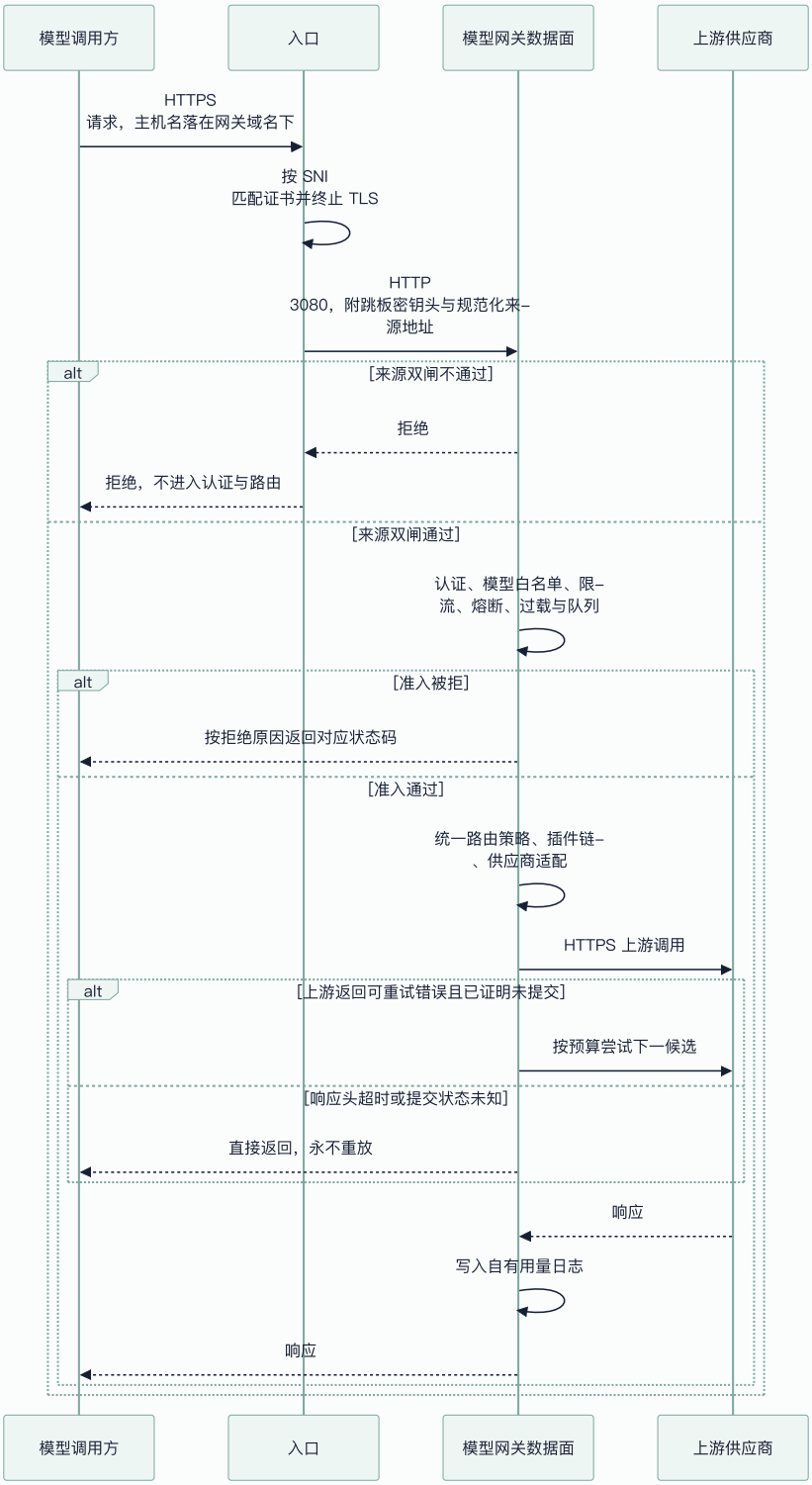


图 2–2　一次模型调用的跳数、认证关口与失败分支

这张图回答的问题是：模型流量在哪一跳被认证、在哪一跳被拒、以及"网关会不会替我重试"这个问题的精确答案。**响应头超时属于"提交状态未知"，永不重放、永不切换候选**；第 3 章会把这条语义与另外两种"切换"分开写清。

> **说明 · 护栏的判定位置**
>
> 护栏的判定位置是**控制面的对话路径**与**受治理调用执行面的 MCP 路径**；模型网关数据面这条链路上的判定是来源校验、认证、准入、流量保护与供应商解析这几层。需要对模型流量做内容治理时，走的是控制面侧的对话路径，见第 4 章。

2.4.3 受治理能力调用链路

外部 MCP 客户端与连接器调用走的是另一条路。控制面对外提供受治理的 MCP 端点与配套的凭据发放、吊销接口；替换版入口配置为这条路径保留了流式转发与 OAuth 发现端点。

跳	协议与端口	凭据与判定位置
外部 MCP 客户端 → 入口	HTTPS 443，流式转发	分发凭据由控制面按 MCP 服务发放
入口 → 控制面	HTTP 3000，容器网内	跳板共享密钥头
控制面 → 受治理调用执行面	HTTP 3021，容器网内	由主密钥派生的服务令牌，在传输层校验
受治理调用执行面 → 能力平面	HTTP 3020	高风险与写操作的复核；未配置该地址时高风险动作 fail-closed
受治理调用执行面 → 护栏	HTTP 3050	调用方密钥；该链路 fail-closed
受治理调用执行面 → 外部系统	HTTPS，按能力配置	外部凭据在本服务内解密，不回传给调用方

表 2-5　受治理能力调用链路的逐跳凭据与判定位置

三个细节决定了这条链路的性质。第一，执行面的调用接口有**传输层的服务令牌闸**，未持令牌的请求在到达请求体里的租约与身份校验之前就被拒绝，这条路径不存在"未认证可达"的探测面。第二，**外部系统的凭据在执行面这一跳被解密并派发，调用方始终拿不到明文凭据**。第三，请求与响应都过护栏，且护栏不可用时拒绝执行而不是放行。Zentrix 的护栏档位装配安全词库与规则式个人信息识别两类引擎，分档差异见第 4 章。

2.4.4 计量与告警链路

计量服务以固定周期、按游标向模型网关数据面拉取用量日志（每页默认 100 条），投影到关系库的用量表；产品侧的用量与成本口径读投影表。模型网关数据面自有的用量日志是一手真相，投影是二手。

告警服务承担规则评估与事件投递两件事，其工作开关与投递开关在出货编排中默认为真。事件经通知适配器翻译成各渠道的报文格式后发出；渠道配置由控制面写成一份只读投影文件，适配器只读该文件，配置本身不进 API 响应。

失败分支：计量拉取失败时投影停在上一游标位，不会写入半截数据；告警投递失败由适配器侧的健康检查暴露，而不是静默堆积。

2.5 入口面与站点划分

Zentrix 使用一份专为网关形态编写的**替换版入口配置**，它是一份独立文件而不是在基线配置上做增删。该配置声明四个站点：

站点	承担什么	后端
`https://*.{网关域名}`	模型调用的对外面，按 SNI 匹配证书	模型网关数据面 3080
`http://*.{网关域名}`	明文入口，供数据面自己执行 HTTP 层策略判定	模型网关数据面 3080
主站点	控制台静态资源、平台 API、受治理 MCP 端点与 OAuth 发现	控制面 3000
指标站点	入口自身的指标端点	入口进程

表 2-6　替换版入口配置声明的四个站点

模型网关数据面的 gRPC 平面（3090）在单机编排中没有入口转发规则，也不发布宿主端口，只在容器网内可达。 它的历史内嵌身份鉴权模式已被废弃，配置到该值时进程直接拒绝启动，而不是降级运行。

替换版配置还在边缘把若干接口直接置为 404：接口文档与规范文件、平台内部 MCP 接口、告警接收接口、数据面注册令牌签发接口、控制面指标端点、技能注册表。这些接口的后端本身仍有各自的鉴权闸，边缘 404 是纵深防御而不是唯一防线。

网关对外面的证书匹配同样是确定的：入口只按已装载的证书匹配 SNI，证书由平台写入一个只读卷、由证书重载器监视并触发热重载，因此换证不需要重启入口。

2.6 网络分区与暴露面

单机编排声明三张容器网络，Zentrix 的实际占用情况如下：

网络	默认网段	在 Zentrix 形态下承载
infra	`10.244.1.0/24`，动态分配区间 `10.244.1.128/25`	平台服务与全部有状态存储
agents	`10.244.2.0/24`	九个平台服务在此网上声明第二张网卡
metrics-discovery	未声明固定网段，声明为内部网络	内部网络，无外网出口

表 2-7　三张容器网络在本形态下的承载

宿主暴露面按三档划分：

暴露方式	服务
宿主全部网络接口	**仅入口的 80 与 443**（可用 `LP_CADDY_HTTP_PORT` 与 `LP_CADDY_HTTPS_PORT` 覆盖）
仅绑回环地址	控制面 3000、运行时数据面前置 3010、能力平面 3020、受治理调用执行面 3021、告警 3022、计量 3023、维护 3024、关系主库 5432、内存态 6379、对象存储 9000 与 9001、列式库 8123
完全不发布	授权服务、连接池、密钥服务、护栏、**模型网关数据面**、通知适配器

表 2-8　宿主暴露面的三档划分

入口在 infra 网上另有一个静态地址（默认 `10.244.1.10`），它同时是模型网关数据面可信代理白名单里的唯一条目——换句话说，数据面只承认从这一个地址过来的转发头。

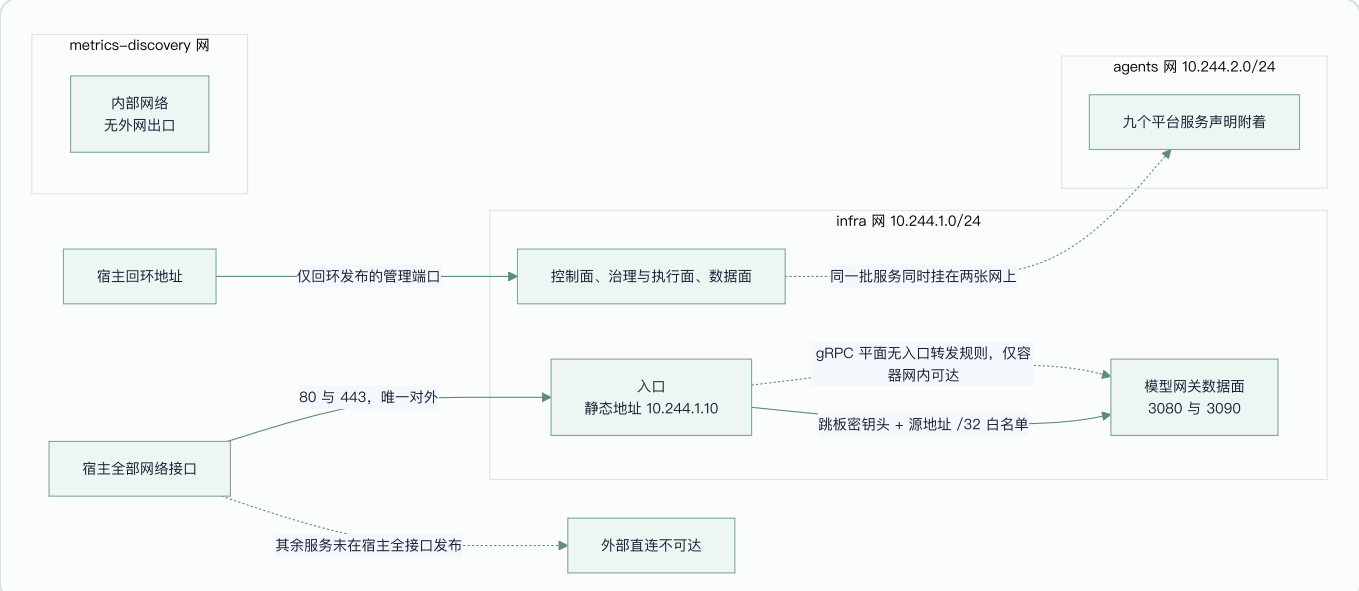


图 2-3　Zentrix 的网络分区与端口暴露面

这张图回答的问题是：防火墙上到底需要放行几个端口，以及模型网关数据面为什么在容器网内也不是"谁都能连"。

> **能力状态**
>
> "唯一对外端口是入口的 80 与 443"在 Zentrix 形态下是**默认启用**的编排事实：其余全部服务的端口映射要么绑定回环地址，要么根本不声明映射。

2.7 运行时配置代际闸

这是一条架构级的一致性设计，它解决的问题不是"配置写错了"，而是"配置和二进制不是同一代"。

每个服务的环境变量并不直接来自运维手写的文件，而是来自一份**按服务隔离的编译投影**：一次编译产出一代投影，编排文件里每个服务只读到属于自己的那一份，且该文件被声明为必需。编排文件顶部另有三个标签，用 fail-closed 语法声明当前运行代际、登记表摘要与编译器版本——**未编译配置代际时，这三个变量为空，一个容器都创建不出来**，而不是带着不完整配置起来。

在此之上还有一道摘要比对：**一个一次性前置校验服务在任何服务接触数据库之前运行**，把镜像内嵌的登记表摘要与本地投影摘要做比对，不一致就让启动失败。它只读挂载安装目录、不接任何网络、失败即终止而不重试。平台库迁移、网关库初始化与密钥库迁移这三个前置任务，都以"该服务成功完成"为启动条件。

> **说明 · 为什么需要这道闸**
>
> 用几天前编译的配置投影去喂新版本的二进制，故障现象通常是这样的：某个明明已经在登记表里声明过的变量，在新二进制里读不到，进程起来又崩、崩了又被拉起，形成一个完全指不到根因的崩溃循环，而报错位置离真正的原因很远。代际闸把这种故障压缩成一句可读的失败信息，代价是每次改配置都必须重新编译一代。

> **能力状态**
>
> 编排文件顶部的代际标签与一次性前置校验服务在 Zentrix 交付包中**均为默认启用**：该校验服务已列入本形态的服务白名单，三个前置任务对它的启动条件依赖在渲染后仍然成立。

2.8 这对你意味着什么

如果你是要把 Zentrix 接进现有网络的架构师，本章可以收敛成四件事：**对外只需为入口放行两个端口**，模型网关数据面自己不发布任何宿主端口，能连到它的只有入口那一个静态地址；**模型流量的可用性不被管理面绑架**，数据面只等控制面启动、不等它健康，护栏也不在这条链路上；**凭据的解密位置是明确的一跳**，受治理调用的外部凭据在执行面解密、不回传调用方；**配置与二进制的代际由编排层强制对齐**，这决定了你的变更流程必须包含一次编译动作，而不是直接改文件重启。剩下的判断题只有一个：这台机器要多大。第 6 章会说明本产品为什么用标定框架而不是推荐配置表来回答它。

第 3 章 · 模型网关核心机制

3.1 为什么模型调用需要独立的一层

让每个业务进程各自持有供应商密钥、各自直连公网端点，会同时丢掉四件事：调用主体不可辨认，故障不可定位（是被限流、是供应商故障，还是自己发错了协议），成本不可归因，策略不可变更。这四件事一旦丢掉，就再也补不回来——事后没有任何一条日志能证明某一次超额支出是哪个团队、哪个应用、哪条路由产生的。

Zentrix 把这四件事收进一层独立进程，即**模型网关数据面**。它是整套部署里所有模型流量的唯一入口，这一点决定了它的全部设计约束：某一处判定的失败姿态选错，影响的不是一个功能，而是全部租户的全部模型调用。本章讲清这条链路的判定顺序，并把三个被广泛滥用的词——"自动切换""每分钟限流""熔断"——在本产品里的精确语义还原出来。

Zentrix 交付包共 31 个服务，其中三个进程的名字最容易与模型调用链路混淆，本书对它们的写法固定如下。

本书统一写法	进程	在 Zentrix 形态下的职责
模型网关数据面	zstack-router	承载全部模型调用：地址与域名归一、认证、准入、路由、插件、供应商适配、上游调用与用量记账
运行时数据面前置	lp-gateway	持有控制面签名下发的策略快照并维护本地台账。它的模型出口路径以智能体身份为前提，不在本章的模型调用链路上
受治理调用执行面	lp-agent-gateway	单次受治理能力（受治理 MCP 与连接器）调用的执行位点：校验冻结的会话能力快照、自行解密凭据、经护栏与流量保护，高风险动作回能力平面复核

表 3-1　三个易混称进程的职责划分

该表回答的是"同一套部署里出现三个代码名带 router 或 gateway 的进程时，哪一个才是模型流量经过的那一个"。答案是第一行，其余两行不在模型调用链路上。

三者的暴露面也不同：模型网关数据面在标准编排中不发布任何宿主端口，外部流量经唯一南北向入口终止 TLS 后反代进入；运行时数据面前置与受治理调用执行面只绑回环地址，对外的受治理 MCP 流量由入口经控制面转入执行面。受治理调用执行面在本形态下的主用法是能力平面治理链路：凭据不下发给调用方，而是在执行面内解密后直连上游。

3.2 一次模型调用要经过的七层判定

3.2.1 前三层：地址、域名与身份

第一层归一客户端地址。 可信对端范围在代码中默认只有回环地址（`127.0.0.0/8` 与 `::1/128`），标准编排把它收窄为入口容器的单个 `/32` 地址。非可信对端送来的转发类请求头一律被剥离，传输协议按明文认定，而不是听信请求头的自述。

第二层是域名网关。 主机名小写并去端口后归一；域名与绑定关系的投影每 1 秒在一个可重复读事务内整体刷新到进程内快照，避免读到半张表。传输策略有三种取值：放行明文、重定向到加密、拒绝；非可信代理送来的明文请求在非放行模式下直接被拒并单独计数。对"未注册域名"的关闭式拦截由 `ZR_GW_BASE_DOMAIN` 启用：把网关基础域名配进这一项之后，该域名下未注册的主机名在域名层即被拒绝。

第三层是认证。 凭据有三种呈递形式，其中 Basic 形式解出的用户名必须与密钥标识常量时间相等才继续。校验按固定顺序进行：密钥哈希常量时间比对 → 密钥状态 → 有效期（有效期字段存在但无法解析为时间戳的同样按已过期处理）→ 地址白名单（IPv4 与 IPv6 网段）→ 所属用户与组织的启用状态 → 只保留启用的角色集。**认证结果不做缓存，每次请求都回数据库核验。** 控制台会话身份不能用于数据面，会被以"需要 API 令牌"拒绝。

3.2.2 第四层准入：各道闸的固定顺序

准入层的判定顺序是固定的，不是可配置的：模型白名单 → 每调用方令牌速率 → 统一路由策略求值 → 过载 → 通用速率算法（排队与默认结果是它的命中动作）；熔断另有固定的求值相位，在端点选择之后（见 3.8 节）。

模型白名单在任何路由工作发生之前评估，密钥无权访问该模型时返回 `403`，错误码 `model_access_denied`，并留下策略决策记录。每调用方令牌速率紧随其后：超限返回 `429`（`token_rate_limit_exceeded`），而**计数后端不可用时返回 `503`**——这一支是关闭式的，它宁可拒绝也不放行。密钥快照上的限额优先于数据库联表结果，目的是让控制面的修改立刻生效而不必等一次联表竞态。

统一路由策略在此之后求值。再往后的过载闸不由策略驱动，它按主机 CPU 与内存水位判定，阈值出厂即有默认值（见 3.8 节）；通用速率算法与熔断则是策略驱动的：控制面没有下发对应动作的编译快照时直接放行，因为"没有策略"不构成拒绝的理由。

3.2.3 后三层：插件、协议与上游

第五层是插件链（见 3.9 节），第六层是供应商适配与协议变换，第七层是上游调用与用量记账。用量在网关自有的关系库中落地，随后由计量进程按固定周期拉取投影（见 3.11 节）。

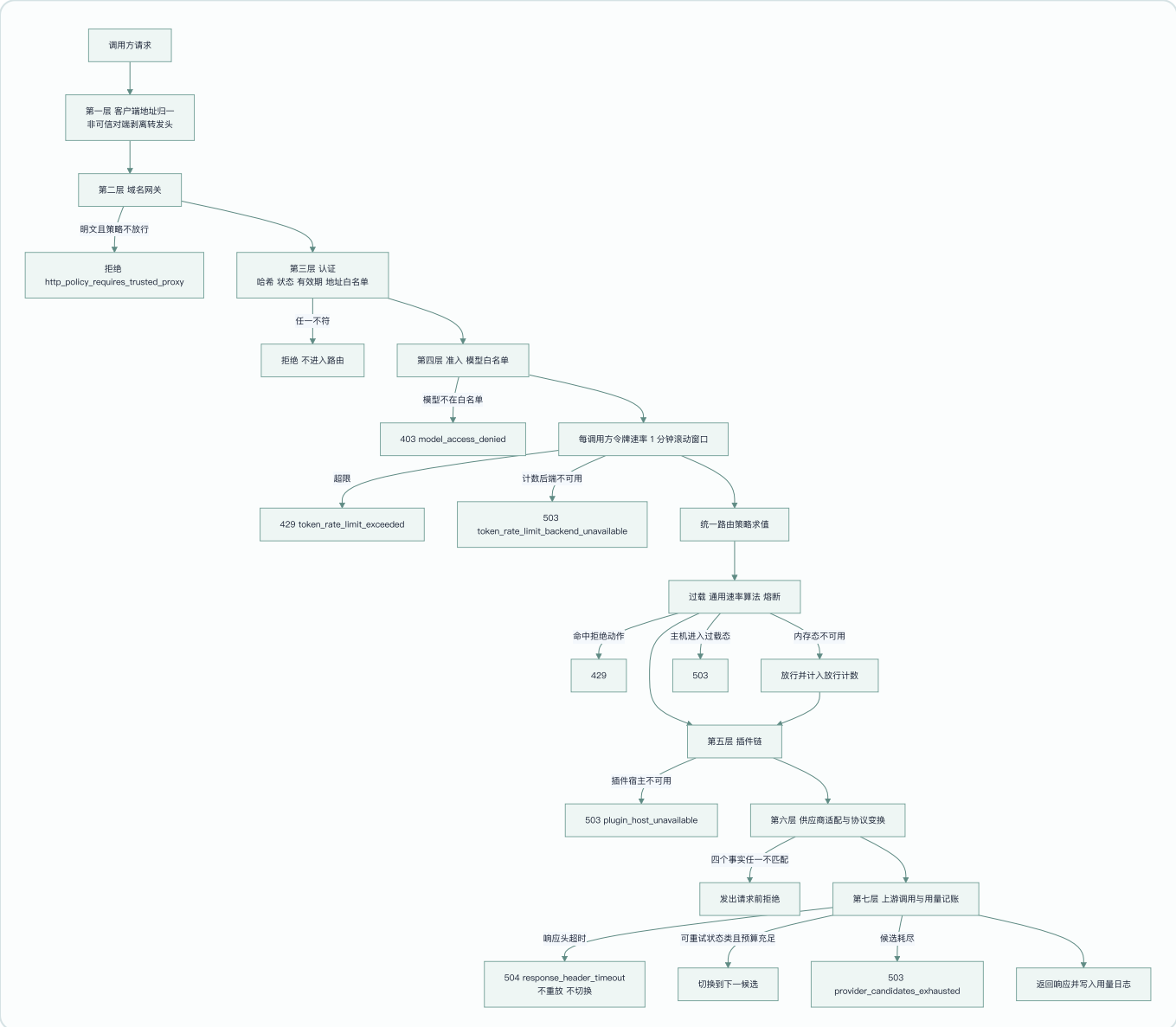


图 3–1　一次模型调用的七层判定链

该图回答的是"请求在哪一层被拒绝、拒绝时返回什么"，因此每一层都画出了失败分支与对应状态码。

3.3 供应商抽象：四个事实必须同时精确匹配

为什么需要。 接入模型供应商最常见的做法是"填个地址、猜个协议、跑通为止"。这种做法在第一家供应商上省下几分钟，在第五家上开始出现难以复现的偶发错误——同一个模型名在两家供应商上语义不同，同一个基址在灰度期指向两套后端，而错误路由的代价是发错供应商、算错成本、把数据送到不该去的上游。

怎么做的。 网关内置 **54 个供应商标识**加 1 个可配置的通用供应商，全部是编译期常量而非数据库表——增删供应商需要重新构建二进制。线路协议是 **17 种取值的封闭枚举，其中 5 种可用于流式响应**；协议层强制的请求头只有一条，即 Anthropic 消息协议固定携带版本号 `2023-06-01`。

运行时解析供应商使用四个事实作为联合键：**供应商标识、精确的生效版本、端点键、网关端点类型**，四者必须同时精确匹配，否则在发出任何上游请求之前失败。契约逐条禁止五类兜底：按渠道声明的协议类型推断、按目录适配器类型推断、按基址域名推断、按模型名推断线路协议，以及"未知供应商默认按 OpenAI 兼容方式调用"；别名解析同样被禁止。

> **注意**
> 供应商目录里的默认基址只是管理台的填写便利值。运行时一律使用渠道自己持有的基址，目录值不作为源权威。

适用范围与参数上限。 通用供应商用于接入 OpenAI 兼容的自建端点，硬上限包括配置 JSON 最大 64 KiB、端点最多 16 个、模型规则最多 256 条、最大超时 300 秒、词元上限 1 000 000；可用的鉴权方案 4 种，只接受 POST 方法，模型匹配使用 ASCII 通配语法。上游调用的 TLS 信任锚是操作系统根证书库，且这是显式声明的依赖特性而非其他组件的传递副作用。

这对你意味着什么。 客户自建的、由内部私有 CA 签发证书的模型端点可直接接入，不必为网关单独准备信任库；但接入任何一家新供应商都必须显式声明四个事实，没有"填个地址就能跑"的路径。这是一次性成本，换来的是长期不会发错供应商的确定性。

3.4 对外端点面与两套入口

网关端点类型是一个 11 项的封闭枚举，其中 10 项可路由，另有模型列表。**当前基线上已实现并可对外调用的是 9 个调用端点与 1 个模型列表端点**：对话补全、响应式对话、Anthropic 消息、文本补全、向量化、图像生成、语音合成、重排序、词元计数，以及从租户模型路由投影生成的模型列表。

入口有两套，认证体系不同。网关原生入口使用网关自有的密钥体系，密钥上直接携带模型白名单、速率限额、队列优先级与地址白名单；平台兼容入口在路径上区分 OpenAI 与 Anthropic 两种形态，使用平台侧租户与工作空间的网关模型密钥体系，密钥行上带有效期、吊销时刻与吊销原因、绑定的模型与计费关系、声明的客户端类型，另有两个仅限服务令牌访问的路由策略求值与模拟端点。平台兼容入口的 HTTP 客户端**显式禁用自动重定向**：出网校验以单个源为单位批准，自动跳转会让已批准的公网源转到未批准的内网或元数据地址。禁用重定向本身无条件生效，与出网守卫是否启用无关。

> **能力状态**
> 流式重放与断点续传由 `ZR_STREAM_REPLAY_ENABLED` 启用，同时配置一把不少于 32 字节的续传密钥；语义续写由 `ZR_SEMANTIC_CONTINUATION_ENABLED` 启用，以流式重放已启用为前提。gRPC 入口平面在标准编排中监听 3090，只在容器网内可达（REST 平面不受影响，未设 gRPC 监听地址时进程按纯 REST 提供服务）；gRPC 反射通过 `ZR_GRPC_REFLECTION` 启用——反射会在任何凭据校验之前回答"这里有哪些服务"，因此它属于按需在内网调试时才打开的开关。

3.5 "切换"的三层语义

"备用模型"这个说法可能指以下三件完全不同的事。它们的触发条件、可见性与安全含义都不一样，混为一谈会直接导致故障预期错误。

3.5.1 第一层：意图路由的兜底服务

虚拟模型必须配置一个兜底模型服务。三种情况会落到它：意图集未启用、没有规则命中、分类器不可用。**兜底一定可见**——决策结果携带"已使用兜底"标记并投影到接口回执，代码里有专门的测试断言兜底不能是静默的。兜底服务同样要通过可见性检查，不在可见集合内时直接以 `400` 拒绝，而不是绕过授权去用它。

3.5.2 第二层：模型服务的主模型不会静默替换

模型服务的主模型严格取配置中的第一位。**主模型被禁用时网关直接报错，绝不静默切到下一个已启用的模型**，且错误信息必须点明是主模型被禁用；没有对话用途配置时同样报错而不是崩溃。这一条由测试固定，测试名本身就是"主模型被禁用时报错而非静默切换"。

> **注意**
> 如果某处文案写的是"主模型不可用时自动切到备用模型"，那是错的。第二层没有自动切换；有自动切换的是第一层的意图兜底与第三层的上游候选，两者的触发条件与第二层完全不同。

3.5.3 第三层：网关侧的候选切换只在"证明未提交"时发生

一条模型路由可以有多个有序候选目标，网关按顺序尝试。切到下一个候选需要同时满足四个条件，缺一不可：

1. **上游确实返回了响应头**，且状态码属于可重试类：`408` 记为超时、`429` 记为限流、`500` 至 `599` 记为服务端错误。**其余状态码（包括绝大多数 4xx）一律不重试，原样返回给调用方。**
2. 该错误类在**这个目标自己配置的可重试集合**里。默认四类全开（传输、超时、限流、服务端错误），但运维可按目标收窄；一旦把限流移出集合，`429` 就直接返回，候选目标一次都不会被调用。
3. 尝试预算还有剩余：总尝试次数未达上限，且当前目标还有重试次数或还有下一个目标。
4. 退避时长装得下路由的整体耗时预算；装不下就把当前上游响应原样返回。

两个绝不切换的反例是安全语义，不是缺陷。 其一，响应头超时属于"提交状态未知"——墙钟超时可能发生在请求字节已送达供应商而响应头尚未返回之时，没有传输里程碑就没有证据，因此返回 `504`（`response_header_timeout`）并且永不重放、永不切换。其二，切换指标只有一个原因标签，取值就是"已证明未提交"，即这条路径在设计上不存在第二种切换理由。

派发前被判定为不健康或熔断已打开的候选会被整体跳过，这不计为切换；全部候选在派发前都不可用返回 `503`（`NO_HEALTHY_MODEL_TARGET`），提交前全部失败返回 `503`（`provider_candidates_exhausted`），整体耗时预算耗尽返回 `504`。

传输超时是四段独立值，默认连接 10 秒、响应头 30 秒、流空闲 60 秒、单次尝试合计 300 秒，且必须满足连接 ⩽ 响应头 ⩽ 单次尝试的顺序约束，数据库侧对这四列有同样的范围与顺序约束。重试默认 2 次，初始退避 250 毫秒、退避上限 2 秒。

> **说明**
> 路由整体的尝试预算，契约允许范围是 1 至 16 次尝试、1 秒至 1800 秒耗时。**当前基线内存在两组不同的生成默认值**（3 次 / 120 秒 与 2 次 / 60 秒），分属不同的规则生成路径，因此本文不给单一数字：实际默认为 2 至 3 次尝试、60 至 120 秒整体预算，取决于该条路由规则由哪条路径生成。

3.6 统一路由策略：可配置的部分与刻意不可配置的部分

统一路由策略是网关侧唯一的声明式分流与改写机制，**策略驱动**：租户没有发布策略时它不参与判定。匹配条件有 **12 类**：路径、请求头、查询参数、灰度 Cookie（只传摘要）、客户端网段、分类标记、个人信息类型、数据来源、模型、调用方、流量标签、时间窗口；同一条规则内的条件之间是"与"关系。比较运算符分成两组且**故意不相交**：单值匹配器只有精确、前缀、正则三种，标签匹配器只有等于、属于、不等于三种。

动作包括加权目标、请求变换、响应变换、结构化字段变换、请求与响应头改写、状态码映射、插件绑定、降级条件与原因码。目标选择策略有加权哈希与平滑加权轮询两种，**必填且没有默认推断**：加权哈希由逻辑请求样本确定性映射，幂等重放会选中同一目标；平滑加权轮询的游标是每个网关副本本地的，模拟接口只窥视不推进它。权重分流发生在端点健康与候选切换判定**之前**，它不是端点级高可用机制。

三条边界必须写清：**响应变换只覆盖非流式的缓冲 JSON 响应，流式路径明确不在范围内**；响应变换在结算之后执行，因此计费永远看到真实的上游响应体；**降级判定同样只在非流式路径求值**，命中时触发降级切换事件并向调用方返回 `502`。事实归一化另有硬上限：单个事实取值不超过 160 字节、单个维度取值数不超过 32 个，个人信息伞标记只要检出任意一项就一定携带且永不因截断被淘汰。时间窗口条件必须使用请求携带的求值时刻，实现中禁止在求值处读取当前时间，否则同一次请求在重放与模拟时会得到不同结论。

3.7 流量保护（一）：三套限流与两类配额

为什么不合并。 同一套部署里有三套彼此独立的限流实现。它们的作用域、时间语义与失败姿态本就不同，强行合并会迫使其中最关键的一层接受错误的失败姿态。

实现	作用域	窗口语义	失败姿态	启用方式
每调用方令牌速率（模型网关数据面）	单个调用方密钥	真·1 分钟滚动窗口，同时统计请求数与词元数	计数后端不可用即拒绝（`503`）	在密钥上开启并设定上限后生效，逐密钥配置
通用速率算法（模型网关数据面）	策略声明的复合主体	周期是显式枚举：秒 / 分钟 / 小时；配额桶按日历窗口重置，时区固定且不可配置	内存态不可用即放行，计入独立计数器	策略驱动：控制面发布含对应动作的策略后生效
HTTP 中间件（控制面）	租户 / 工作空间 / 用户	1 秒桶（字段名含 rpm 但语义是每秒）；词元上限另按分钟键统计	内存态未配置即报错拒绝	默认启用，总开关 `RATE_LIMIT_HTTP_DISABLED`（未设即启用）

表 3–2　三套限流的作用域、窗口语义与失败姿态对比

该表回答的是"某一处限流失效时，请求会被放行还是被拒绝"。

两处语义差异必须单独强调。其一，"每分钟"在三套实现里含义不同：网关每调用方令牌限流是真正的 1 分钟滚动窗口；通用速率算法把周期保留为显式枚举，"每分钟 1 次"不会被折算成有损的整数秒速率；而控制面 HTTP 中间件里名为 rpm 的字段实际是 1 秒桶。其二，**网关侧的通用速率闸在内存态不可用时刻意放行**——它是所有模型流量的唯一入口，一次内存态抖动会同时让所有租户的所有调用返回 `503`；补偿手段是独立的放行计数器，而真实的限流拒绝仍然返回 `429`。控制面则相反：词元上限准入在内存态未配置时直接报错，因为它影响的是单个能力而不是全部流量。控制面侧默认值是租户与工作空间各每秒 100 次请求、每分钟 100 亿词元；后者在代码注释中写明是刻意调到"实际等同不限"，让限额成为选择开启项而不是选择关闭项。

配额是另一类东西，与速率无关。通用速率算法带日历配额桶：周期枚举为小时、天、月与终身，按日历窗口整体重置而不是滑窗，时区固定且不可配置，其中终身周期使用调用方自有的代际而不是墙钟。多个桶同时参与时必须走多桶原子提交——先全量判定再统一提交，避免"被拒的请求已经烧掉前面几个桶的额度"。

3.8 流量保护（二）：命中动作、熔断、过载与并发

命中动作有三种：拒绝（默认值，立即 `429`）、排队、默认结果，多条策略同时命中时裁决顺序固定为拒绝 > 排队 > 默认结果。默认结果返回 HTTP `200`，但**不调用上游、不计费、用量为 0**，结束原因是一个独立取值而不是正常完成，原因码上限 64 字节、内容上限 65 536 字节。排队模式有四个上限：每策略队列深度默认 100、上限 512；最长等待默认 5 秒、上限 30 秒；每组织在队请求数 128；每组织字节预算 32 MiB。等待时长超过上限时什么都不提交，既不写状态也不推进计数。队列优先级只信任密钥上的绑定、绝不接受客户端请求头，取值 0 至 3、默认 0。计数维度是显式声明的复合主体，**声明的维度在请求上缺失时该规则不适用，而不是塌缩到更粗的桶**。

熔断是滚动窗口失败率加慢调用率的三态实现（闭合、半开、打开），状态存放在内存态中因而跨副本共享，半开探针带租约，求值相位固定在端点选择之后。契约七个字段各有范围：窗口 1 至 3600 秒、最小调用数 1 至 1 000 000、失败率与慢调用率各 1% 至 100%、慢调用时长 1 至 1 800 000 毫秒、打开时长 1 至 3600 秒、半开最大调用数 1 至 1000。旧的"连续失败计数"字段已删除且没有保留别名——旧载荷会被拒绝，而不是被按新语义重新解释。**熔断是策略驱动的：由控制面发布含熔断动作的策略后在数据面生效。**

过载保护是基于主机 CPU 与内存的三态滞回门：CPU 进入阈值 85%、退出阈值 70%，内存进入 90%、退出 80%，恢复态停留 30 秒，采样周期 5 秒；处于过载态时新请求返回 `503`。在主机级过载门之外，进程级（按 cgroup 归一化）的 CPU 与内存门通过显式配置阈值启用，让容器自身的水位也参与过载判定。上游在途并发上限由 `ZR_UPSTREAM_MAX_IN_FLIGHT` 设定（取 0 表示不设上限），超出上限的上游调用先等许可，许可等待默认 100 毫秒。

网关策略集有四个策略族：请求阻断、防重放、词元上限、并发上限，各自有独立启用位，按需逐族启用。并发上限的默认配置是限额 1、键域为调用方、超限只有拒绝一种动作、租约 300 秒；策略集规范 JSON 上限 256 KiB。并发软池必须显式开启，否则软硬限字段保持惰性，避免升级时静默改变既有部署的上限；软硬限翻转发生在内存态的原子脚本内，越过软限后新请求被压回软限，**在途请求不会被杀掉，靠自然排空**。

3.9 插件机制：数据面上可编程的扩展位

为什么需要。 企业接入模型时总有几条规则是产品无法预置的：某个部门的请求必须改写模型名、某类调用要在进入路由前先问一次自有的额度服务、某个回复流里的特定字段要就地重写。把这些规则做成产品配置项会无限膨胀，做成分叉代码会让升级失去意义。Zentrix 的做法是在数据面上开出固定的扩展位，让这些规则以受限的、可撤销的、带资源上限的方式运行在热路径上。

怎么做的。 插件是编译为 WebAssembly 组件的独立制品，通过控制面签名下发的插件平面进入数据面，而不是由运维往目录里放文件。插件平面携带六类内容：插件目录、制品（带 SHA-256 摘要、字节大小与目标架构）、安装（启用位、缺件处置策略、执行模式、来宾配置、生命周期代际）、绑定（作用域、执行模式、采样率、优先级、启用位、修订号）、密钥绑定（逻辑名、密钥引用、**允许的目的地**）以及每个安装自己的网关策略集。密钥以引用加允许目的地的形式绑定，插件配置里不出现明文凭据。

插入相位共五个，按流水线顺序执行：模型改写（在模型白名单之前，可改写请求的模型名）、请求闸（路由选择之前，可拒绝请求）、路由后闸（路由选择之后，可基于选中的路由再拒绝一次）、请求变换（发出上游调用之前，可改写请求体与请求头）、响应变换（流式响应侧的逐块改写）。响应侧的插件链最多 3 个。

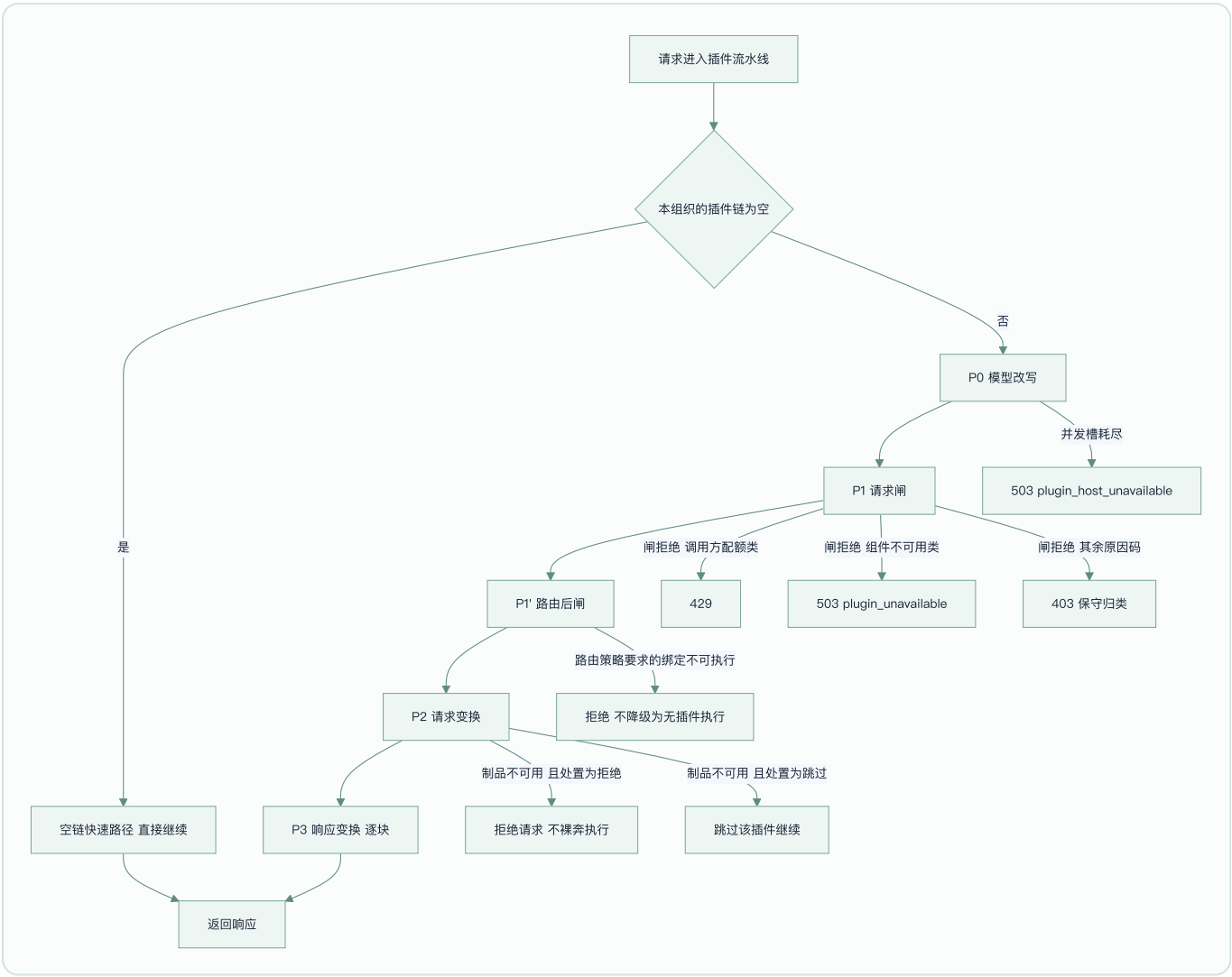


图 3–2　插件链的五个插入相位与失败分支

该图回答的是"插件坏了会怎样"——注意它有三条互不相同的拒绝出口，这正是本节最重要的设计。

边界与默认状态。 插件沙箱的限额被刻意压到通用 WASM 沙箱的百分之一量级：每次执行燃料 2 000 000、线性内存 4 MiB、计算超时 20 毫秒、最大实例数 10、最大表数 10；原因写在代码注释里——数据面插件跑在热请求路径上，必须以微秒计完成而不是毫秒。每次请求使用全新的执行存储而不复用，因为执行存储是只增的资源竞技场，复用到第 11 次就会撞上最大实例数。并发槽在进程级按池容量封顶，单个组织最多占用池容量的四分之一，槽耗尽返回 `503` 而不是 `403`。

拒绝原因的三分法是一条被测试钉死的纪律：调用方配额类返回 `429`，插件宿主不可用类返回 `503`，其余未知原因码一律保守归为 `403`。注释写得很直白——把"安全插件坏了"报成 `429`，既撒谎，又会被下游按"限流"洗白成"模型调用过于频繁"，这是最坏的组合。

执行模式有强制与只观测两态；制品不可用时的处置策略有三种：拒绝请求、跳过并计错、跳过且不计错。统一路由策略若声明了必需的插件绑定而该绑定不可执行，请求被拒绝，不会降级成"无插件执行"。**未绑定任何插件的组织走空链快速路径**，一次原子读即返回，这是"开箱不装插件不产生代价"的实现方式。

这对你意味着什么。 你可以把自有的准入规则、字段改写与协议适配放进网关热路径，而不必分叉产品代码；代价是这些规则必须能在 20 毫秒与 4 MiB 内完成，并且要为"插件不可用"显式选择一种处置策略——这个选择不给默认值，是因为"安全插件失效时继续放行"与"业务插件失效时拒绝全部流量"都是真实存在的错误答案。

3.10 内容护栏的判定位置

护栏的判定位置是控制面的对话路径与受治理调用执行面的 MCP 路径。 模型网关数据面这条链路上的判定是前面几节讲的准入、路由、流量保护与供应商解析；需要对模型流量做内容治理时，走控制面侧的对话路径。把护栏画进模型网关数据面的调用链是常见的误读，做安全评估时请按上面这条口径核对。

护栏服务本身在 Zentrix 包内**默认启用**：编排上不受任何档位门控，随栈常驻运行。它装配哪几类引擎由能力档位文件决定，**Zentrix 形态装配安全词库与规则式个人信息识别两类引擎**，档位差异见第 4 章表 4–3。

3.11 计量与成本口径

为什么需要一手真相。 成本归因如果只读二手汇总，任何一次投影中断都会变成"账对不上但查不出差在哪"。Zentrix 的做法是让用量日志落在网关自己的关系库里作为一手真相（也可以指向一个独立库），再由计量进程以固定 **15 秒**周期、按游标分页拉取（每页 100 条），投影到平台关系库的两张表：滚动词元汇总表与带维度的成本归因事件表。投影带有游标状态、认领记录、死信与毒信轨道，单条最多重投 20 次。

产品用量与成本口径一律读这两张投影表；列式库在当前基线上已降级为原始观测用途，代码注释明确指出成本真相不在列式库。计量契约本身也是版本化的：用量来源是一个封闭枚举，结果是三态（成功 / 失败 / 不确定）——"不确定"是一等取值，而不是被折算成成功或失败；金额以整数微单位加币种表示，且微单位不得为负。三态与整数金额共同保证"上游状态未知"在账目上可见、可对齐，而不是被四舍五入掉。

> **必须知道的两处放行**
>
> 其一，签名配置载荷里的成本预算平面在当前基线上是**告警性质**：本地闸检出超预算会计入独立计数器，真正拦截仍由同步计费预留完成。其二，计费服务慢或不可用时，请求被本地闸**放行且不计费**，同样计入一个独立计数器。这是"可用性优先于精确性"的显式取舍，补偿手段是这两个计数器必须被纳入监控——它们持续上涨就意味着账目正在失真。

> **能力状态**
>
> 模型网关数据面持续暴露低基数指标端点，可由客户自有采集系统抓取；租户与工作空间标识明令不得作为指标标签，需要按租户、工作空间或调用方归因时走本节这两张用量投影表。链路标识由进程自行生成并随请求贯穿传递，用于把同一次请求在指标、事件与日志之间对齐；把这些信号接进客户既有观测体系的方式见第 8 章。

3.12 控制面与数据面的配置契约

为什么需要。 数据面的域名、路由、策略与插件配置如果来自本地文件的手工编辑，就会同时出现三种事故：两侧对同一个字段的理解不一致、旧格式配置被按新语义误读、控制面临时不可达时数据面不知道该守住旧配置还是清空。

怎么做的。 这些配置一律来自控制面编译并签名的配置载荷。签名算法为 Ed25519，覆盖载荷的规范 JSON 编码；双方必须序列化同一个契约结构，这个共享契约存在的唯一目的就是消除两份手抄定义。**当前唯一被接受的载荷版本是 6，且刻意没有兼容窗口**：一份格式合法的旧版本载荷会在反序列化阶段直接失败，而不是被按新语义误读。载荷携带域名拓扑、路径绑定、每租户模型路由、成本预算、流量治理编译快照、插件平面、统一路由策略、网关参数与签名密钥标识。其中"未分配"标志位是防误删的关键：控制面配了选择器但本实例未被匹配时，各平面为空且该标志置位，应用方**不得**据此裁剪已有的配置行。

生命期是两套：配置载荷生存期 30 天、达到三分之一时重铸，模型路由投影租约同为 30 天且续期条件是"控制面可达且仍在背书"（经过认证的"未修改"响应同样续租）；健康快照最长 120 秒，且从编译签发时刻起算而非应用时刻起算，为控制面刷新与数据面轮询各留余量。数据面回报的已应用修订号只有配置与健康两个键，另附积压、时钟偏移、策略决策、流量策略副本状态与插件平面状态。

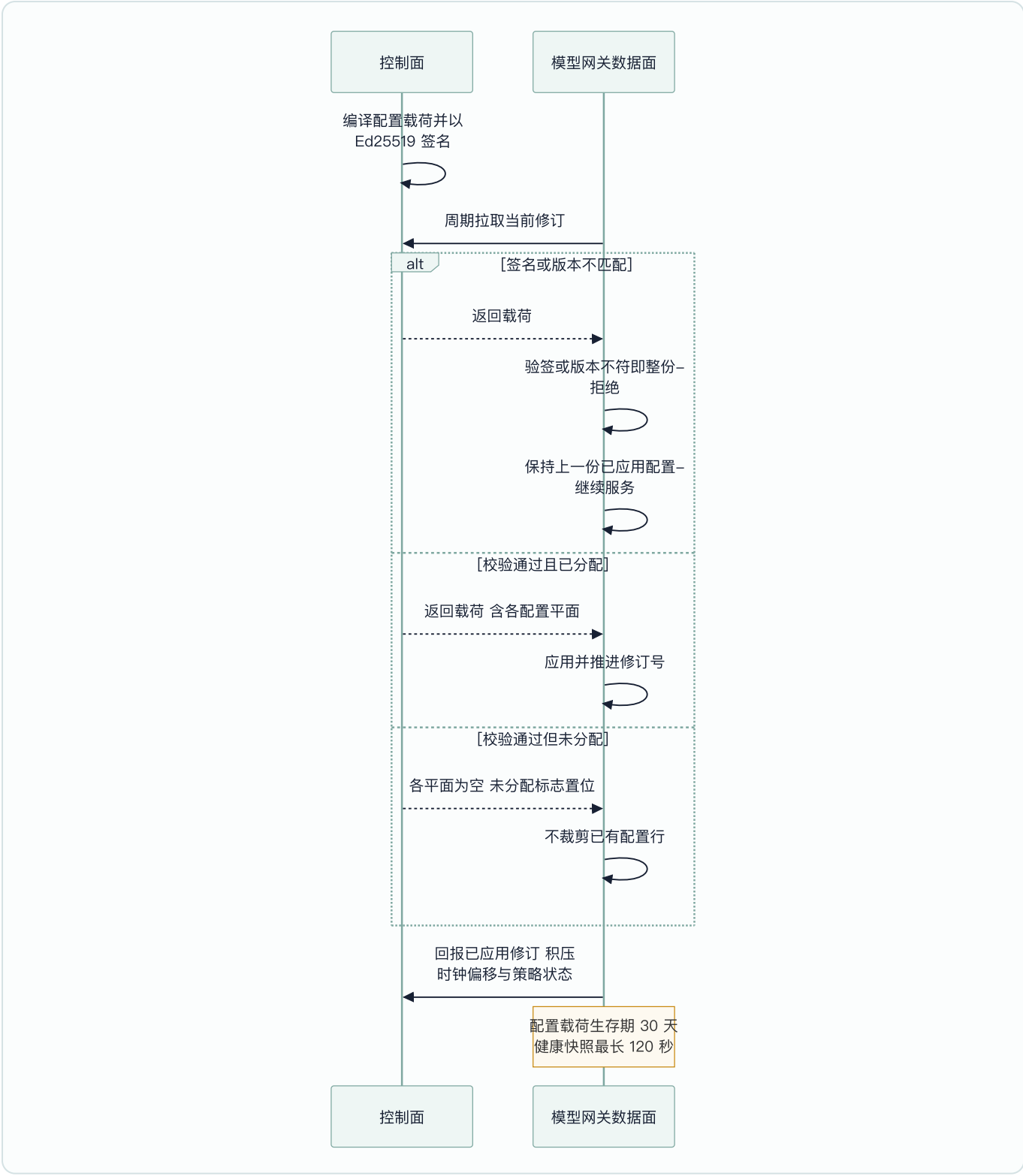


图 3-3　控制面签名下发与数据面应用回报

该图回答的是"控制面不可达或下发了不合法配置时，网关会怎样"，因此画出了拒绝与未分配两条非正常分支。

这套契约在部署侧还有一道前置保障：Zentrix 包内含一个一次性的**配置代际校验**服务，它在数据库迁移之前运行，不联网、只读地比对镜像内嵌的配置登记表摘要与本地配置投影的摘要，比对不通过则整条启动链停在这里。编排文件顶部另有三个用关闭式语法声明的代际标签，未编译配置代际时一个容器都创建不出来。两者合起来消除的是同一类事故：用旧配置投影喂新二进制，症状是完全指不到根因的崩溃循环。

运行时数据面前置侧还有一道网关证明校验，强制开关 `GATEWAY_ATTESTATION_ENFORCE` **默认为真**，只有显式关闭时才把未通过证明的请求降级放行并记入审计。该进程在编排上刻意只等待控制面"已启动"而不是"已健康"，并可在冷启动时

从磁盘加载经验签的快照先行服务。真正把模型流量的可用性与管理面的可用性分开的，是模型网关数据面自身同样只等控制面"已启动"这一条。

3.13 这对你意味着什么

如果你负责把模型调用接进现有系统：接入新供应商需要显式声明四个事实，没有"填个地址就能跑"的兜底路径；扩展逻辑走数据面插件而不是分叉代码，代价是要为插件不可用显式选一种处置策略。如果你负责稳定性：请按三层"切换"分别写应急预案，尤其记住响应头超时既不重试也不换目标；同时记住网关侧的通用速率闸与计费闸在依赖抖动时刻意放行，对应的两个计数器必须进监控。如果你负责合规与成本：用量与成本以网关一手日志经固定周期投影的两张表为准；通用速率限流、熔断与统一路由策略都是策略驱动的能力，由控制面发布对应策略后在数据面生效，因此上线前要把"发哪些策略"当成一次明确的配置动作排进交付计划。

第 4 章 · 安全架构

4.1 把"安全"拆成六条可验证的控制面

安全评估的难点不在于产品宣称做了什么，而在于评估人能否逐条验证"在什么条件下，哪一个组件拒绝了什么"。本章因此把安全拆成六条可分别取证的控制面：它们的判定都发生在请求期或进程启动期，由数据库约束、进程内中间件或启动自检强制。通过开关启用的控制面一律写明开关名与启用后的影响——只说"能力存在"而不说怎么开，会让评估人把"能力存在"读成"防护已经生效"。

控制面	覆盖范围	启用方式	开关或边界
关系型授权	19 类对象，含平台域、租户域、工作空间、供应商配置与网关模型	默认启用	授权服务缺席即拒绝启动；不可达一律拒绝
密钥与信任根	集群密钥、每租户数据密钥、服务间令牌	默认启用	主密钥是唯一必填密钥，不可原地轮换
租户数据隔离	关系库行级安全 + 应用层谓词静态检查	默认启用	会话变量未设置即返回 0 行
网络与宿主边界	端口暴露面、跳板认证、出网守卫、宿主容器套接字	暴露面与跳板认证默认启用；出网守卫按客户出网策略启用	平台侧 `ssrf_guard_enabled`；模型网关数据面侧 `ssrf_settings.enabled`
内容安全与个人信息保护	安全词库、规则式个人信息识别	引擎默认启用；22 条内置个人信息规则按合规要求逐条启用	每条规则各有启用位；护栏地址未配置即拒绝启动
审计与留痕	写操作审计、动作目录、哈希链、导出	默认启用	审计事实默认保留 730 天

表 4–1　六条控制面的启用方式与开关

> **说明**
>
> 本章出现三个"面"，职责互不重叠，请按名区分：**模型网关数据面**承载全部模型调用流量；**运行时数据面前置**持有由控制面签名下发的策略快照并维护本地台账，其三组路由以智能体身份为前提；**受治理调用执行面**负责单次受治理调用的准入执行与凭据解密，外部 MCP 客户端的受治理调用走这一条。三者都不是"入口"——唯一南北向入口是边缘反代。

4.2 关系型授权：判定在请求期发生

4.2.1 授权模型与它的变更约束

权限权威是一套关系型授权服务，采用关系推导而不是权限位表：19 类对象定义写在仓库内的单一模型文件里，随控制面二进制一起发布。控制面启动时读现网模型做差分迁移，迁移前先取分布式锁，已有实例正在迁移则报错退出。模型变更中只有"删除一个关系"被判为破坏性操作：它必须出现在编译期白名单里，否则启动阶段直接拒绝，当前白名单仅两项——权限模型的收缩无法在一次常规升级中被静默完成。

作为授权的前置条件，认证侧的参数同样固定：访问令牌 3600 秒（1 小时），刷新令牌 30 天且宽限期外的重放会撤销整条链，登录失败限速 5 次 / 60 秒；管理员代登录令牌 180 秒，有全局急停开关 `impersonation.disabled`（默认未设置，即代登录可用），开始与结束各写一条同步落库的审计。

4.2.2 角色分立：租户三员与平台四角色

租户侧实装等保 2.0 的三员分立：配置管理员、安全管理员、审计员三者互斥，已持有其一的用户不能再被授予另一个，互斥由授权写入路径强制；另有运维员角色不参与互斥。四个角色均为内置角色，授予关系只存在于授权服务中，关系库里没有影子表可绕过。

平台侧另有四个角色加一个超级管理员，两处设计值得注意：平台四角色**不**隐含租户管理权，只有超级管理员经"平台管理"关系传递到租户管理权——否则平台审计员会一次性拿到全部租户的管理权限；平台角色的主体类型只接受用户、不接受用户组，防止租户内的组成员身份成为跨租户提权通道。

> **注意**
> 护栏证据（脱敏前原文）的读取权限是全模型中唯一不包含平台超级管理员的租户权限：只有租户管理员与租户安全管理员可读，平台超级管理员在该端点上被拒。这不是配置，是权限模型本身决定的。

4.2.3 判定链、缓存与不可达时的姿态

执行面与安装面的判定强制使用强一致读，调用方要求更弱的一致性也会被提升；只有浏览类查询可以在进入降级档后使用低延迟一致性，而降级档有两条入口：由运维开关 `SPICEDB_CONSISTENCY_DEGRADE` 显式开启，或分发对账积压时自动进入。开关的每次翻转都写一条审计级别的日志，自动进入这条路径不写。进程内只缓存**肯定**结果，容量 10000 条、生存期 5 秒；否定结果不缓存，避免新建资源尚未传播时被长期判定为不存在。浏览面另有一份"最后已知良好"快照，容量 50000 条、生存期 600 秒。授权客户端自带熔断：错误率阈值 0.5、闭合窗口 50 次、半开探针 5 次、打开后等待 30 秒。

授权服务不可达时的行为最需要逐条核对：租户域与平台域的权限闸一律返回 503，从不放行；浏览面可服务那份快照，但它只含肯定结果、冷条目仍然隐藏。代码里确实存在一个自动放行的分支，但它只在桌面单用户形态下对租户闸生效，服务器形态没有这个逃生口，Zentrix 按服务器形态出货；平台域闸连桌面形态都没有这个口子，唯一的例外是平台超级管理员校验。调用方密钥这类非用户主体在租户权限闸上直接被拒、不做往返——该闸的主体类型只接受用户与用户组成员；拒绝响应体不含权限域与权限名，无权者拿不到权限地图。

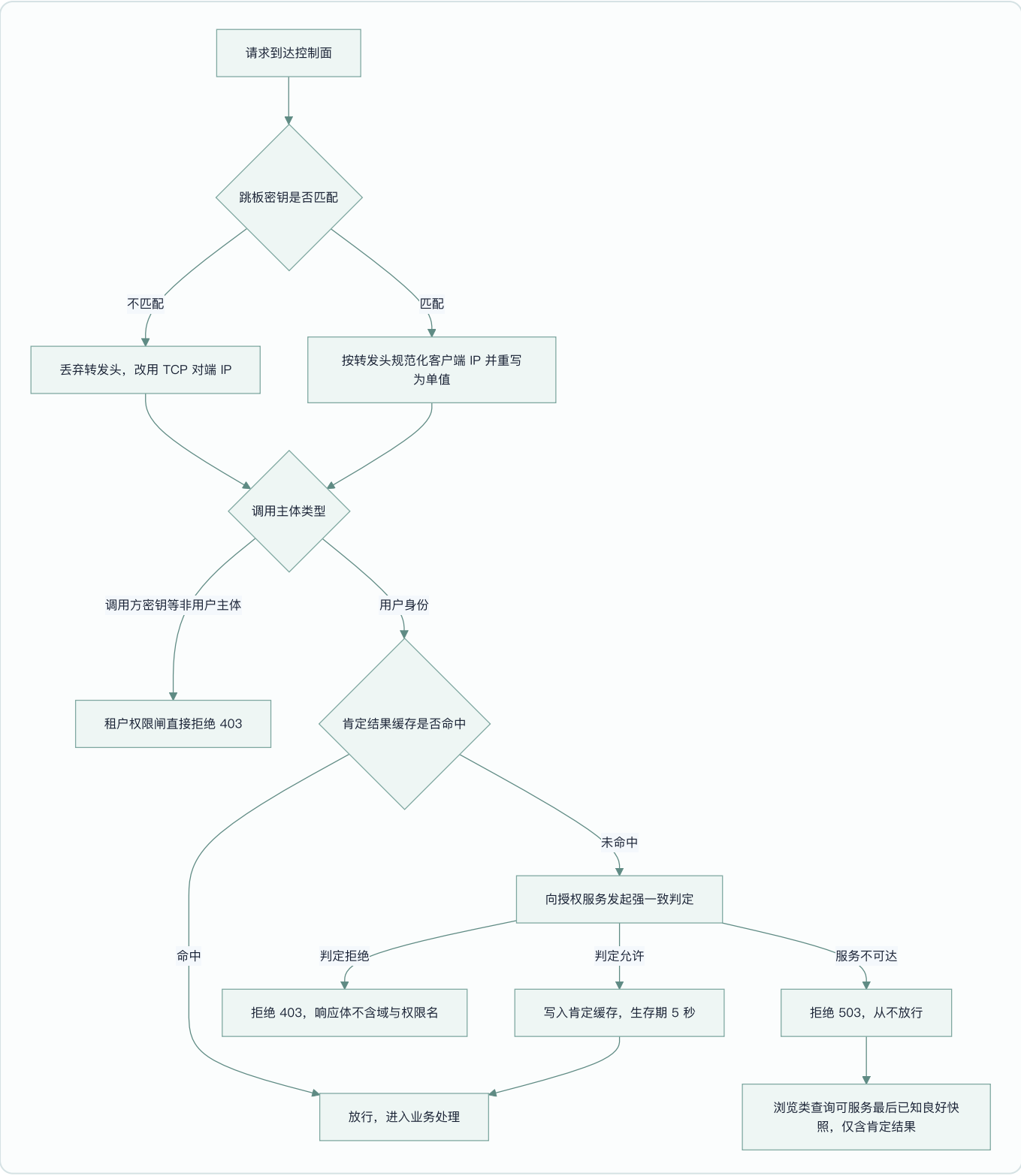


图 4–1　一次受保护请求的授权判定链

这张图回答"什么条件下请求被拒绝、拒绝码是什么"，授权服务不可达的分支刻意画成拒绝而非放行。

4.3 密钥与信任根

4.3.1 一把主密钥派生出的东西

平台自身的密钥体系只有一个信任根：32 字节的部署主密钥，也是运维必须自行提供的两个必填项之一（另一个是对外访问地址）；唯一不挂在它下面的是独立密钥服务（见 4.3.3）。其余密钥全部由它派生：34 种集群密钥经 HKDF–SHA256 在固定标签下派生，覆盖各数据库角色口令、会话签名密钥、跳板密钥、服务间令牌与浏览器推送密钥对；每

租户数据加密密钥由主密钥与租户标识派生，用于以 AES-256-GCM 加密供应商凭据、受治理调用的绑定配置与脱敏原文证据。派生集合是编译期固定的，不随交付形态增减。

数据库、授权服务、对象存储、入口这些第三方容器无法自行派生，因此安装期把派生结果物化成一份权限 0600 的只读文件，文件头写明"由派生器生成、请勿编辑"与主密钥指纹（主密钥的 SHA-256 值）。指纹让备份清单、界面展示与安装校验回答同一个问题：运行中的系统与这份备份是不是同一个信任根。

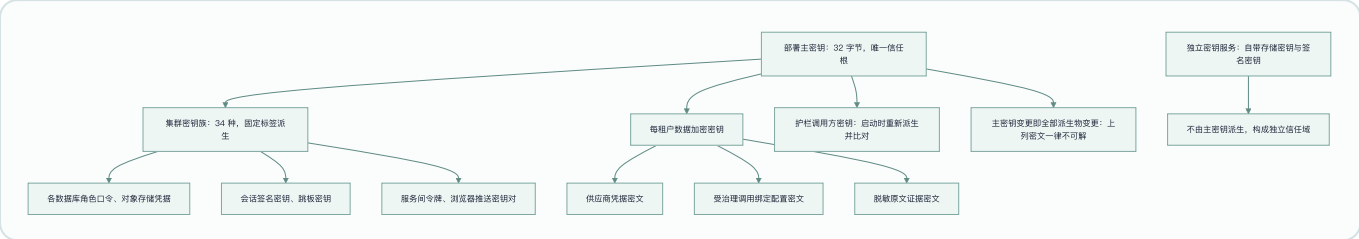


图 4-2　主密钥派生树

这张图回答"丢失或更换主密钥会连带失去什么"，右侧的独立密钥服务说明并非所有密钥材料都挂在同一个信任根上。

4.3.2 为什么不可原地轮换，代价是什么

主密钥**不提供原地轮换**。Helm 交付路径检测到既有密钥属于不同信任根时直接失败；单机形态在指纹变化时告警，说明所有派生密钥都已轮换、既有浏览器推送订阅全部失效需重新订阅。原因是结构性的：全部集群密钥与每租户数据密钥都从这一把密钥派生，允许原地轮换等于允许信任根在运行中分叉——数据库口令、授权服务令牌、对象存储凭据一次性全部改变，而已落库的密文仍属于旧根。

代价必须一并写清楚：主密钥保管成为一等运维职责；主密钥丢失意味着 34 种集群密钥与每租户数据密钥都无法重算，供应商凭据密文、受治理调用绑定配置密文与脱敏证据密文一律不可解（此时证据被显式标记为不可解，而不是当作空记录或报错）。缓解手段只有指纹校验与显式失败，绝不在指纹不一致时"尽力继续"。

> **高危**
> 更换主密钥不是一次配置变更，而是一次信任根迁移：必须按"停机—以新根重新派生全部密钥—重新加密或接受既有密文不可解"处理。产品不提供把旧密文迁移到新根的原地路径。

4.3.3 独立信任域与三道启动期闸

独立密钥服务。 它自带存储加密密钥与授权签名密钥，不消费部署主密钥；权威服务只持有加密与签名材料以及一份**公开**的工作负载信任包，工作负载私钥留在各自的轮换器里，对权威服务与同机其他进程都不可见。它签发的工作负载令牌生存期 120 秒、每 30 秒轮换，生存期硬边界 30 至 300 秒，越界即拒绝启动；控制面与模型网关数据面各持一套独立签名身份，两者的密钥标识与公钥必须互不相同，Zentrix 包为此带了两个独立的令牌轮换器。

护栏调用方密钥自检。 护栏启动时用部署主密钥重新派生两把调用方密钥，与实际加载的密钥串做常数时间比对，任一不符即**拒绝启动**。它针对的是一个具体的升级事故形态——编排文件默认指向仓库内的开发密钥文件，跳过配置编译步骤的升级会带着众所周知的密钥上线。检查放在启动期，这种错误就不会以"看起来正常运行"的方式存在。

配置代际校验闸。 每个服务读到的环境都是一次编译产物，由运维手写项、派生密钥与镜像内嵌的配置登记表共同生成。这道闸是一个一次性前置容器：不连网络、以只读方式看一眼安装目录，比对镜像内嵌登记表摘要与本地投影摘要，不一致就让整次启动失败，并在一句话里同时给出两个摘要。它刻意不做成别的服务的健康检查——它必须在任何进程碰数据库之前跑完，且必须让启动失败而不是让系统降级运行。三条迁移入口（平台库、模型网关库、密钥服务库）都以"该闸成功完成"为前置条件；POC、完整形态与 Zentrix 三个交付包都带这道闸。它消灭的事故形态是：用旧投影喂新二进制，运维看到的是一个指不到根因的崩溃循环。

4.4 租户数据隔离：两道彼此独立的闸

第一道闸在数据库内：81 张表启用并强制行级安全，其中 68 张有策略、共 119 条；另有 13 张**不设任何策略**——对结构上没有租户列、写不出谓词的表，做法是保留强制行级安全且完全不设策略并回收请求角色的表权限：请求角色因默认拒绝而一行都读不到，这比写一条"日后可能被放宽"的谓词更难被误改。

策略谓词统一是"租户列等于事务级会话变量"。会话变量未设置时谓词求值为空，结果是返回 0 行而不是返回全部——这是隔离设计里最容易被写反的一处。读写策略分离，写策略带检查子句，拒绝把外租户标识写进本租户上下文。角色矩阵同样是硬约束：请求路径角色不具备绕过行级安全的属性，只有后台系统角色具备，且持续集成禁止两者互相授予；另有六条不变量在活库上执行，其中一条要求策略快照与登记快照逐字节一致，任何漂移都会绊线。关系库与连接池两侧的口令散列算法都固定为 scram-sha-256。

第二道闸在应用层：一套租户谓词静态检查棘轮，要求每条查询要么自带租户条件，要么带逐条的作用域标注，否则必须显式登记在基线中；当前基线 871 行，更新时只做交集——只能删除、不能新增，新违规必须手工写进文件并留在改动记录里，整文件级的豁免标记已经退役。两道都要不是冗余：行级安全是运行期的最后一道防线，静态检查棘轮则让"新代码有没有带租户条件"在合并之前就可判定。对象存储侧的隔离粒度是每租户一个存储桶，下发的是限定到该桶的管理凭据，评估时应按"桶级隔离"记账。

4.5 网络与宿主边界

4.5.1 暴露面与跳板认证

整套编排中唯一在宿主全接口发布端口的服务是入口（80 与 443）；其余发布端口一律绑定回环地址，授权服务、护栏、密钥服务、连接池、通知适配器与模型网关数据面根本不发布宿主端口，只在容器网内可达。内部网络上运行的是平台自身的这批服务，因此容器网内的可达关系可以逐个服务地枚举与核对。

入口到控制面之间的信任由一个共享密钥跳板头承载。这里有一处流传很广的误述必须纠正：**它是共享密钥的常数时间比较，不是每请求的 HMAC 签名**。密钥由主密钥派生，运维不手工设置；启动时做硬校验——长度须在 32 至 256 之间且全部为可见 ASCII 字符，命中三个公开占位值之一或与会话签名密钥相同均拒绝启动。运行期判定同样明确：同名头出现多次即视为不匹配；不匹配时丢弃转发来的客户端 IP 头、改用 TCP 对端地址；匹配时按固定优先级取第一个可解析地址，再把这几个头全部删除并重写为规范化的单值。伪造客户端 IP 因此需要同时拿到跳板密钥。模型网关数据面一侧是双闸：既校验跳板头，又用可信代理网段限定源地址只能是入口容器的 /32；其自带默认网段在容器网桥下永远匹配不到，因此该值必须在部署时显式给出。Zentrix 的入口配置在网关子域的明文与加密两个站点上都注入跳板头，两条路径的判定条件一致。

4.5.2 出网 SSRF 守卫：按客户的出网策略启用

守卫本身是完整的：连接期重新解析并重放拒绝名单（关闭 DNS 重绑定窗口）、解析结果固定到具体地址、禁止跟随重定向、已验证解析缓存 300 秒最多 1024 条、单个地址最多固定 16 个、DNS 查询超时 2 秒最多 2 次尝试。拒绝类别包括云元数据地址、私网段、回环、链路本地、唯一本地地址、运营商级 NAT 段，以及解析到上述任一地址的主机名，拒绝返回 400。

它由平台运行配置中的全局键 `ssrf_guard_enabled` 启用，只能由平台凭证域管理员经管理接口翻转，翻转后跨进程传播间隔 10 秒。**这是一个整体开关**：启用后上面整条地址拒绝名单在各执行链上生效；未启用时保留一条最低限度校验——地址必须可解析且协议必须是 http 或 https。企业内自建的模型端点与对象存储大量位于私网，因此这项守卫的正确交付动作是：先与客户确认出网策略与内网白名单范围，再在上线前开启。各条执行链的覆盖需要分开看：

执行链	守卫形式	启用方式
平台 HTTP 连接器与动作执行	连接期重解析 + 地址固定	随全局开关 `ssrf_guard_enabled` 生效
受治理调用执行面转发到外部 MCP 服务	同上，另加禁止重定向、请求超时 30 秒 / 连接超时 5 秒	同上

执行链	守卫形式	启用方式
MCP 注册与健康探测、能力包导入	写入前校验 + 探测前解析校验 + 地址固定客户端	同上
模型网关数据面调用上游模型	数据面自有出网设置，按同一套拒绝名单判定	数据面设置项 `ssrf_settings.enabled`（设置项，不是环境变量）；未启用时仍要求协议为 http/https 且主机非空
MCP 调试用的内部客户端	直连平台内部回环服务，不经守卫	既定设计的例外，受治理的转发路径不走这条

表 4–2　出网与转发链路的守卫覆盖与启用方式

平台内部服务之间的调用另有一份精确放行清单：16 个"主机:端口"条目，仅精确匹配、无通配，畸形条目被静默丢弃而不是宽松解释；该清单是编排文件里的固定值。外部注册中心同步来的上游端点使用另一份默认为空的清单，且刻意不与内部清单合并，避免被恶意注册的端点折叠到平台内部服务上。

> **说明**
> 浏览器侧默认注入禁止内容类型嗅探、引荐来源策略与同源框架限制三类响应头，跨站请求伪造防护采用双提交 cookie 常数时间比对；严格传输安全头只在面向公网的 https 构建下发，内网 HTTP 部署刻意不发，否则浏览器会把该主机锁定到 HTTPS 一年。

4.5.3 宿主容器套接字的暴露面

这套系统的容器编排由控制面自己完成，它因此需要宿主的容器套接字。这一条必须写清楚而不是绕过去：套接字以**读写**方式挂进控制面容器；只读挂载并不构成缓解——只读挂的仍是同一个套接字，Docker API 的创建、执行与停止照样写得进去。

边界因此很窄：**Zentrix 形态下只有控制面一个容器挂载它**，用途收敛为运行期观测与整包升级编排。持有该挂载的容器越少，这条路径的攻击面越小，这也是本形态相对完整形态在宿主边界上的直接收益。

现有的缓解是三条：套接字不出容器、不发布任何宿主端口、也不经网络暴露；能触到控制面容器的前提是先攻破控制面进程，而控制面本来就是主密钥的持有者，二者信任级别相同；控制面镜像与运行时投影都受配置代际校验闸约束（见 4.3.3）。要说清楚的边界是：**这套设计把"控制面被攻破"的爆炸半径定义为整台宿主机**。如果你的威胁模型不接受这一点，正确的做法是把宿主本身当作单一信任域来管（专机部署、限制登录、宿主级审计），而不是指望容器层再挡一道。

4.6 内容安全与个人信息保护

4.6.1 引擎与本形态的交付档位

内容安全由一个独立进程承担，在标准编排里常驻、不受编排开关门控；控制面与受治理调用执行面在未配置护栏地址时**拒绝启动**——对后者来说，外部 MCP 的请求与响应检查是硬依赖，检查不可用即失败关闭，因此地址未配时它宁可不启动，也不留下一个健康检查能过、受治理流量却全部 503 的容器。进程内装配哪几类引擎由交付档位文件决定，档位随包钉死。Zentrix 形态装配以下两类：

引擎	Zentrix 形态状态	依据
安全词库	默认启用	本形态档位标为必需
规则式个人信息识别	默认启用（22 条内置规则按合规要求逐条启用）	本形态档位标为必需

表 4–3　Zentrix 形态装配的护栏引擎

档位文件本身是 fail-closed 的：任何被装配的层都必须带一份钉死的模型工件描述符（引擎标识、版本、精度、摘要），缺一即档位加载失败——这道约束让"以为装了某个模型引擎、其实没有模型文件"无法成立。做安全评估时请按本形态的档位口径核对表 4-3。

安全词库当前冻结为 6 大类、合计 26149 条，来自已核实许可的开源项目并经过分类清洗与误报闸门，定位是冷启动兜底：交付即可用，客户可用自有词库替换或扩展，政务场景以客户提供的权威词库为最终生效版本。

4.6.2 判定、动作与失败姿态

内置个人信息规则共 **22 条**，分四个包（中国核心、金融、医疗、密钥凭据），动作统一为脱敏，**按客户的合规要求逐条启用**——内置包提供的是开箱可用的识别覆盖面，具体启用哪些类别由客户的数据分类分级口径决定。规则加载时只加载已启用的行；三条中国核心规则走归一化加校验位识别，其余为正则；模式长度上限 500 字符且禁止环视等易失控构造。

命中动作四档且严格有序：放行、记录、脱敏、阻断依次趋严，冲突时取更严的一档；**未知动作字符串 fail-closed 成阻断**——策略配置漂移不得把限制性规则退化成放行。只有脱敏及以上的命中参与文本替换，记录档只写违规不改文本。护栏总裁决是三态（放行 / 改写 / 阻断），干预类型四种（已阻断、标准答复替换、拒答替换、流式截断）。规避层归因有两个取值：仅在规范化（去零宽字符、同形字折叠）后命中，或仅在解码（base64、十六进制、URL）后命中，"绕过尝试"与"直接命中"因此在证据上可区分。

治理默认值可逐条核对：个人信息判定开启、降级失败姿态为 fail_closed、模型策略闸开启、私有模型强制开启且强制标签为敏感级与机密级两档。多模态请求里的图像部分在图像流水线未配置时逐个按 fail-closed 处理，不当作干净内容放行。护栏检查本身失败会让整次请求返回 503。数据分级使用四档规范取值（公开、内部、敏感、机密），多标签取最严，未知或遗留标签在分级治理路径上被严格拒绝而非静默降级。

4.6.3 脱敏证据的加密存证与访问边界

脱敏发生时，原文不进入审计表，也不以明文写入任何位置：它被 AES-256-GCM 加密后单独存证，密钥由部署主密钥与租户标识派生，文本与图像（打码缩略图）使用完全相同的密钥与附加数据契约。保留期默认 180 天，取值范围 1 至 3650 天，存放在数据库设置项中（不再读取环境变量），清理器每个周期重读并取全局最小值，因此调整设置无需重启。保留期过后证据行被清理，读取接口返回"原文已按保留期销毁"而不是裸 404，让审计人能区分"没有过"与"曾经有、已按期销毁"。访问边界见 4.2.2：只有租户管理员与租户安全管理员可读，平台超级管理员被拒，且每次读取都写一条审计。

4.7 审计与留痕

4.7.1 采集范围与动作目录

审计中间件默认覆盖四种写方法（POST、PUT、PATCH、DELETE），请求体缓冲上限 64 KiB，超过即不记录请求体，敏感字段在写入前统一脱敏。动作名不是自由字符串：产品维护一份静态动作目录，由持续集成守三件事——目录卫生（非空、唯一、有序）、静态动作棘轮（代码中发现的静态动作必须已在目录内）、直插旁路闸（生产源码不得对审计形状的表直接插入，除非列入允许清单）；中间件按路由动态派生的动作是既定设计的例外。这套机制的价值在于：新增一个不写审计或绕过审计写入的代码路径，会在合并前被拦下，而不是在合规检查时被发现。

4.7.2 不可篡改的两层与它的边界

第一层是审计表上的追加写触发器：更新与删除在触发器内检查当前数据库角色，只有专用的审计维护角色可执行，其余一律以权限不足报错；该角色不可登录，只授予给执行保留期清理与归档的进程。以会话变量作为判据的方案被明确否决，因为会话变量可被伪造。

第二层是哈希链：权威写入方在接受审计事实的**同一事务**内追加一条链条目。链按范围（平台或具体租户）加确定性 16 路分片分区，避免多个独立生产者串行到同一个链头；链条目的哈希值与序号有格式与连续性约束，且其追加写触发器**不看角色**——更新或删除一律报错，连审计维护角色也不例外。保留与导出同样是确定值：审计事实默认保留 730 天

（对齐 ISO 27001 的最低要求）；导出提供 5 个端点，含异步作业与结果下载，导出行为本身写审计，定位是"有界证据导出加显式脱敏"。

关于调用正文还有一条边界值得写明：Zentrix 形态下用量与成本的归因由计量投影表承担，**调用正文不参与归因链路**——投影表里落的是用量、金额与归因维度，不是提示词与回复内容。本形态向外输出的信号出口共三条——指标端点、结构化业务事件与容器日志，逐条口径见第 8 章。

> **说明**
> 哈希链解决的是"事后被改动或被删除能否被发现"，它把篡改从"可能悄无声息"变成"必然可检出"。它不解决、也不宣称解决另外两件事：有写入权限的人选择不写某条记录，以及审计内容本身的真实性。这两件事分别由动作目录棘轮与角色分立承担。

4.8 两条元原则：默认 fail-closed，以及刻意 fail-open 的少数位置

第一条元原则是：不确定时拒绝。它在代码里有一批可逐条验证的实例。

- 授权服务不可达时权限闸返回 503；服务器形态下缺少授权服务，控制面拒绝启动。
- 未配置护栏地址时控制面与受治理调用执行面拒绝启动；护栏调用方密钥自检不通过时拒绝启动。
- 行级安全的会话变量未设置时返回 0 行；13 张无策略的强制表等价于全部拒绝。
- 内存态的淘汰策略不是"不淘汰"时，限流中间件在请求期判定并拒绝服务——被淘汰的限流计数等于限流失效。
- 受治理调用执行面未配置流量保护内存态地址时，每一次受闸的 MCP 调用直接失败关闭；未配置能力平面地址时，高风险与变更类动作的复核器停用，这类动作一律失败关闭。
- 未知的内容策略动作解析为阻断；图像判定链任一步失败按拦截处理。
- 运行时配置未编译时编排文件顶部的必填标签为空，一个容器都创建不出来；配置代际不一致时前置闸让整次启动失败。

第二条元原则是：确实存在刻意 fail-open 的位置，数量很少，每一处都有理由与补偿手段。写出来比声称不存在更有说服力。

位置	为什么刻意放行	补偿手段
模型网关数据面的限流闸在内存态不可用时放行	它是所有模型流量的唯一入口，一次内存态抖动会同时让全部租户被拒	只按"内存态不可用"这一个原因覆盖判定，真实限流拒绝仍返回 429；每次覆盖计入专用的放行计数器
出网 SSRF 守卫在启用之前不限制出网目标（`ssrf_guard_enabled`）	内网自建端点（模型、对象存储）大量位于私网，交付首日即开启会阻断正常连通	启用前仍校验地址可解析且协议为 http/https；开关只有平台凭证域管理员可改，10 秒内跨进程生效；上线前按客户出网策略开启
模型网关数据面自有出网设置在启用之前允许私网、不限端口（`ssrf_settings.enabled`）	同上，且它面对的正是客户自建模型端点	启用前仍要求协议为 http/https 且主机非空；开关是数据面设置项，不是环境变量
MCP 调试用的内部客户端直连回环内部服务	让它经守卫就连不上回环上的平台内部服务，调试面会整体失效	仅限调试客户端，受治理的转发路径不走这条

表 4-4　刻意 fail-open 的位置、理由与补偿

> **注意**
> 浏览类授权查询在授权服务不可达时服务"最后已知良好"快照，这**不是** fail-open：快照只含肯定结果，未缓存过的条目仍然不可见，且执行面与安装面的判定不受该降级影响。评估时应记为"降级可见性"，不要记为"放行"。

4.9 这对你意味着什么

如果你要为 Zentrix 出具安全评估意见，本章给的是一份可逐条核对的清单：控制面的启用方式在表 4-1，出网链路覆盖在表 4-2，护栏引擎的本形态装配在表 4-3，刻意放行的位置在表 4-4。上线前真正需要你做决策的是四件事：是否开启出网守卫、是否为模型网关数据面单独打开出网设置、启用哪些内置个人信息规则、以及主密钥由谁保管与如何备份。前三件是可回退的配置变更，最后一件不可回退，必须在第一次安装之前定下来。还有一件不在开关清单上但必须写进你的评估结论：控制面持有宿主容器套接字，请按"控制面与宿主同信任域"来划边界。

第 5 章 · 数据架构与可观测性

5.1 为什么数据面被切成多块而不是一个库

把模型接进生产之后，Zentrix 里同时存在三类性质完全不同的数据：必须强一致且要做租户隔离的业务状态（调用方密钥、模型服务与路由策略、配额与用量投影、审计事实）、只追加写且按时间过期的观测记录（调用日志、平台日志、健康快照）、以及体积大但访问稀疏的文件（审计归档、租户文件）。它们对一致性、写入放大、索引结构与保留期的要求互相冲突，压进同一个存储会让最严格的那一类被迫接受最宽松的约束。因此数据面被拆成职责互斥的几块，代价是运维面上多了几种存储要备份、监控与升级。本章描述的是 Zentrix 交付形态下的数据面：五类存储、谁直连谁经池、留存多久、怎么演进，以及它向外输出哪些可观测信号。

5.2 存储分工与不合并的理由

存储	承载的数据	为什么不并入主库
关系主库（PostgreSQL 16）	平台业务状态；同一实例上另有两个独立库，分属模型网关数据面与密钥服务	它就是事务与行级隔离的落点；另两个库独立是为了各自拥有 schema 演进节奏
授权服务库（独立实例与独立卷）	关系型授权的关系元组	它有自己的迁移链路，共用实例会让一次迁移同时影响两条链路
内存态（Redis 7）	限流计数、流量保护与熔断状态、外部授权代理会话、运行时配置广播、计量配额	可丢失且写入极频繁，落关系库会污染主库写入路径
列式库（ClickHouse）	平台日志、观测事件、健康快照，以及网关会话推理的原始记录与会话用量	只追加、按列聚合；行存实现会让保留期清理变成长事务
对象存储（S3 兼容）	租户文件本体与审计归档，按租户分桶，平台另有一个平台桶	体积大、访问稀疏；浏览器要能直接取，下载走预签名链接、不经控制面转发

表 5–1　Zentrix 形态下的存储分工

主库容器的共享内存显式设为 1 GiB——容器引擎默认的 64 MiB 会让并行查询直接报错，而且报的是数据库错误而不是资源错误，读者会去查询里找原因。主库宿主端口只绑回环地址；授权服务库与连接池服务完全不发布宿主端口。主库另有一组 Zentrix 专属的启动参数，它们的含义与调整边界放在第 6 章讲，因为那是容量话题而不是数据模型话题。

5.3 连接拓扑：谁直连、谁经池

> **注意**
> 两条流传很广的误述与当前基线不符："控制面经连接池访问主库"，以及"连接池是为执行体做连接复用"。

事实是：**控制面主进程、迁移器、护栏与模型网关数据面全部直连主库**，因为会话级顾问锁需要一条会话连接，经事务级池会失去锁语义；**连接池服务在本形态下的使用者是五个从控制面拆分出去的服务**——能力平面、受治理调用执行面、告警、计量与维护。连接池以事务池模式运行：客户端连接上限 3000，默认池大小 20，通配库的后端连接上限 5、主库单独放宽到 48，后端空闲回收 300 秒、客户端空闲回收 600 秒、后端连接寿命 3600 秒，每次归还执行会话清理。

> **高危**
> 连接容量应按这五个前置服务的并发事务数推算，**控制面主进程的连接不经过池**，改连接池参数不会影响它。任何按"执行体数量乘以每体连接数"推算池容量的口径在本形态下都不成立。

这里也是三个名字需要一次性说清的地方，本书此后严格按这三个名字使用：**模型网关数据面**是承接模型调用流量的进程（需要精确指代时写进程名 `zstack-router`）；**受治理调用执行面**（`lp-agent-gateway`）是受治理 MCP 与连接器调用的单次执行者，它自己在已验证租约上解密凭据，凭据不下发给调用方；**运行时数据面前置**（`lp-gateway`）持有控制面签名下发的策略快照与本地台账。

> **说明**
> 运行时数据面前置在本形态下的职责是**持有已验签快照并具备降级服务能力**：它向控制面登记身份，把经签名校验的快照写进本地台账，进程重启时先从磁盘装载未过期快照再接流量。它同时提供访问准入子请求、平台内部模型出口与服务发现三类内部路由，这三组路由以智能体身份为前提。它在 Zentrix 中随包就位并持续维持快照与台账，加装运行时后立即可用。

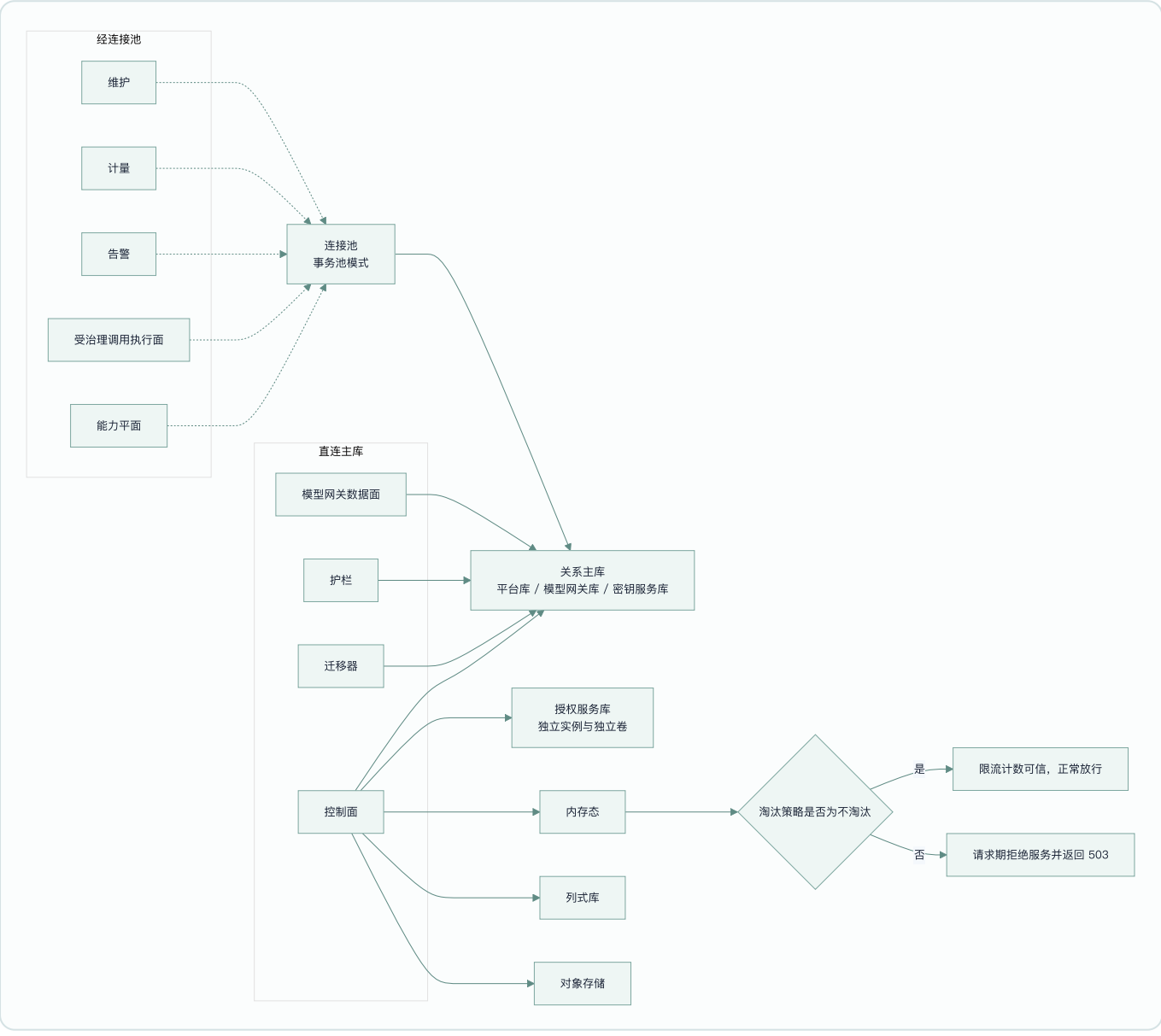


图 5-1　存储分工与连接拓扑

图注：实线为直连、虚线为经连接池。这张图回答两个问题——改连接数配置时应该改在哪一侧，以及内存态配置错了会在哪一刻被拦下。

5.4 单点协调与内存态的硬约束

平台内有一批必须单实例运行的后台任务。**领导者选举走关系库上的可抢占租约表，生存期 45 秒，不走内存态。** 旧版本使用的会话级顾问锁已退役，新副本会主动抢占既有租约。这解决的是滚动升级期间的双主问题：顾问锁在旧连接尚

未断开时不会释放，会让旧二进制在新版本起来之后仍自认领导者。

内存态则有一条不可调优的硬约束：**淘汰策略必须是"不淘汰"**。默认启动参数即为不淘汰，内存上限 1 GiB，开启追加写并每秒刷盘。限流中间件在请求期校验该策略，任何会淘汰键的策略一律拒绝服务并返回 503——限流计数一旦被淘汰，限流就在系统压力最大的时刻静默失效。可覆盖变量是 `REDIS_MAXMEMORY_POLICY`（默认为不淘汰）与 `REDIS_MAXMEMORY`（默认 1 GiB），编排注释明确限定：只有在自带托管内存态并已在别处保证不淘汰时才允许改。

> **能力状态**
> 领导者租约选举、内存态淘汰策略的请求期校验：**默认启用**，无开关。

5.5 迁移工程与两道一致性闸

当前基线的平台库迁移文件共 1252 个：452 个是早期的三位序号编号（最大为 491），800 个是时间戳编号。切换点写死为 491，此后新增迁移必须用时间戳编号；编号 466 到 491 之间那批"未发布但已冻结"的顺序迁移共 26 个被逐条写死在白名单里。原因是迁移执行器以版本号为键记录已应用集合，两个文件撞号时第二个会被静默跳过——事故表现是"某张表在一部分环境里就是不存在"，且没有任何报错。

围绕迁移有三道工程闸，各自的强制点不同，写清强制点比宣称"有闸"更有意义：强制行级安全的表上属主直写必须成对括起，在持续集成上强制；已发布迁移的内容摘要不可变，在离线发版的前置校验上强制；发布风险静态分类把删表、删列、清空、重命名、置非空、新增非空列、唯一索引、约束校验等动作分成"直接拒绝""高风险""需附证据放行"三档，在提交前检查中执行。

运行期还有两道闸，这是 Zentrix 包与 Zenova 完整包同等具备的架构特性。**第一道是运行代际校验**：一个一次性任务以只读方式挂载安装目录、不接任何网络，比对镜像内嵌的配置登记表摘要与本地投影摘要，不一致即让整次启动失败并同时打印两个摘要；编排文件顶部另有三个用失败即停语法声明的标签，未编译配置代际则一个容器都创建不出来。**第二道是 schema 兼容闸**：应用启动时读取数据库已应用的最高迁移版本，与本镜像构建期内嵌的 schema 头版本比对，低于最低支持版本（当前为 465）或高于本镜像已测版本即快速失败，校验查询超时 5 秒；三个边界值各有环境变量可覆盖。

> **注意**
> **控制面的系统连接池刻意不接受这些覆盖变量**，只按本镜像内嵌的头版本判定。这针对一个具体的事故形状：当一个进程的两个连接池指向不同数据库时，被放宽的那一侧会顺利过闸，结果是这个进程带着未经校验的 schema 认知连上了另一个数据库。宽松的一侧让严格的一侧失去了保护，这是配置开关最容易被用错的方式。

5.6 留存与生存期

数据类别	默认保留	调整方式
平台日志、观测事件、健康快照（列式库）	14 天	平台日志留存策略统一改，范围 1–3650 天
网关会话推理用量（列式库）	90 天	固定，不随日志留存策略变动
平台日志总容量	50 GB	`LP_LOG_RETENTION_MAX_SIZE_GB`，默认 50，范围 1–1048576
审计事实	730 天	无
容器日志文件	单文件 50 MiB、最多 10 个	无

表 5-2　留存期与生存期总表

清理动作由独立的维护服务执行（其清理开关在出货编排中为开启；控制面侧的同名开关保持关闭，保证同一份工作只有一个执行者）。清理周期默认 86400 秒（1 天），默认时刻 02:00，时区默认 `Asia/Shanghai`，最小间隔 300 秒；另有一条磁盘压力触发线：剩余空间低于 10 GiB 或低于 10% 时提前触发。本形态下实际执行清理的对象是列式库及其系

统日志表、容器日志、临时目录、审计边界与清理守卫。审计侧的到期处理分批执行，单次不会长时间持有行锁；审计归档的落点是对象存储。

5.7 Zentrix 输出哪四类可观测信号

信号契约把可观测面固定为四类，本形态四类都产出：**指标**由各服务自行暴露拉取端点并要求低基数；**结构化业务事件**以固定字段名写进结构化日志，带独立的事件契约版本号；**日志**由容器引擎按上面的轮转策略落盘；**链路上下文**由进程自行生成并贯穿传递——请求标识、链路标识与标准链路上下文头三者让一次请求在各类信号里可以被对齐，因此即便只收日志，一次请求在各服务之间的关联仍然成立。

> **高危**
> **租户标识与工作空间标识明令不得作为指标标签**（同一条纪律里还列了智能体标识）。网关侧另有测试断言，指标输出中不得出现渠道标识、请求标识与上游基址。这是一条成本与可用性设计：指标标签的基数决定时序库的内存与磁盘占用，一个按租户展开的标签会让指标条数随租户数线性增长，最终拖垮时序库，进而让所有基于它的告警一起失效。

需要按租户、工作空间或调用方归因时，正确路径不是指标，而是另外两条：**用量与成本读关系库里的用量投影表**——模型网关数据面自有的用量日志是一手真相，平台每 15 秒以游标方式分页拉取（每页 100 条）投影到关系库，投影失败的条目进入死信并最多重投 20 次；列式库中的原始推理记录已降级为原始观测，不作为成本口径。**行为归因读结构化业务事件。**

5.8 把信号接进你自己的监控体系

Zentrix 的可观测面按"产出标准化信号、由客户既有体系承接"设计。五条接出路径都是现成的，交付时逐条接通即可：

- **指标**：各服务的指标端点在平台内网与宿主回环上可达，客户自有的指标系统可以直接抓取。抓取点有一条明确约束——边缘对指标路径一律返回 404，抓取必须从平台内网或宿主回环发起，这是刻意的暴露面收敛，把指标面留在内网。
- **日志**：容器日志按固定轮转策略落在宿主上（单文件 50 MiB、最多 10 个），客户自有的日志采集可以直接读。
- **结构化业务事件**：混在日志流里，带固定字段名与事件契约版本号，可以被解析成一条独立的事件流。
- **调用与用量数据**：调用记录与会话用量在列式库，用量成本投影在关系库，都可以用标准 SQL 直接查询。
- **就绪端点**：各服务的就绪端点按八类原因暴露不可用原因（见 6.1），接进客户的巡检体系即可回答"进程活着但不接流量"这类问题。主机与容器层面的 CPU、内存、磁盘、网络指标由客户既有的导出器采集。

> **合规结论**
> Zentrix 形态下可观测信号的默认出口只有三样：低基数指标、结构化业务事件与容器日志，**模型调用正文不在其中**；用量与成本的归因走计量投影表，落的是用量、金额与归因维度。链路标识只用于把同一次请求在这三样信号之间对齐，不携带调用正文；接出方式与本形态的信号出口边界见第 8 章。

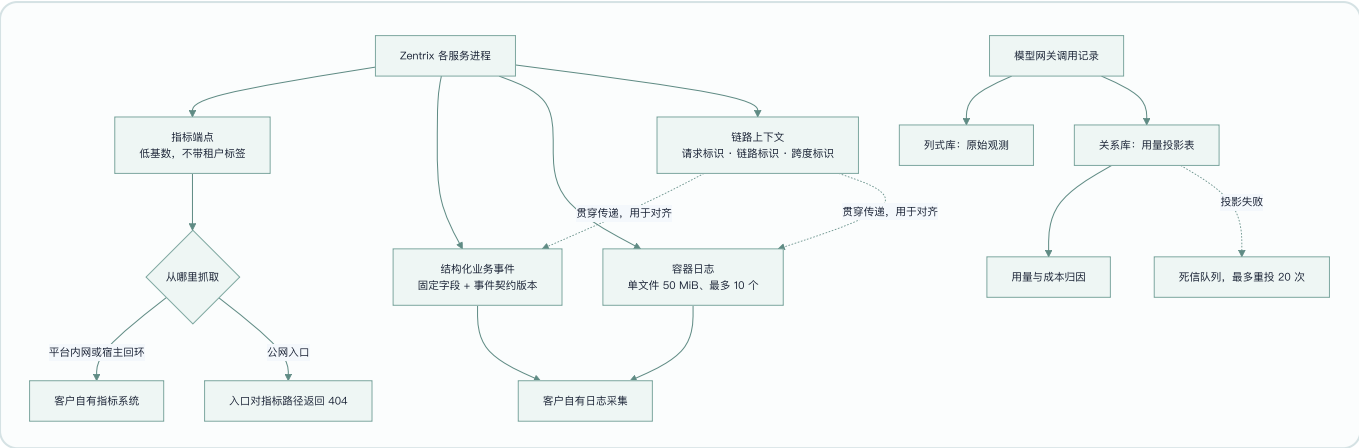


图 5–2　四类信号的产生与接出路径

图注：这张图回答"某一类信号取不到时该先查哪一段"——指标取不到的第一嫌疑是抓取点选在了公网入口，用量对不上的第一嫌疑是投影死信。

5.9 业务告警：随包即用的那一条链路

业务告警链路完整地随包交付：告警规则以数据库为唯一事实源，由独立的告警服务承担规则评估与事件投递，编排中该服务的评估开关与投递开关均为开启；评估按 5 秒一次的节奏复算，用 60 秒的持续窗口判定触发与恢复（避免抖动导致的反复通知），质量类规则另按 30 秒节奏单独求值以免高频重算打到业务库上。

通知出口由通知适配器承担，它把告警信封翻译成钉钉、飞书与通用 Webhook 三种形态，路由约定是按渠道类型与渠道名称寻址；渠道配置由控制面写进一个只读投影文件供适配器消费，凭据不进 API 响应。护栏产生的审计事件也走同一条投递链路：告警服务作为账本消费者读取护栏审计事件并按投递计划发出，该消费循环用独立的租约保证同一时刻只有一个副本推进投递水位。

> **能力状态**
> 业务规则告警与通知投递：**默认启用**，随包即用。基础设施层的指标告警与可视化看板接客户既有监控体系，接法见 5.8。

5.10 这对你意味着什么

如果你是企业架构师，这一章给你的判断依据是：本形态有四类独立存储需要各自的监控与升级窗口（内存态按可丢失设计，其余三类进入备份集），连接容量按五个前置服务推算而不是按调用并发推算，迁移是单向的且有三道强制点不同的工程闸加两道运行期一致性闸。如果你是安全或合规负责人，需要落进评估意见的有三条：审计事实默认保留 730 天且归档落在对象存储、调用与日志类数据默认 14 天（会话用量 90 天）、以及模型调用正文不参与用量与成本的归因链路。如果你要把 Zentrix 接进既有监控体系，需要在验收清单里写明的是：指标从内网抓、日志从宿主收、成本从用量投影表读、就绪端点接进巡检，基础设施层的采集与看板由你既有的体系承担。

第 6 章 · 可靠性、容量与性能

6.1 存活与就绪是两件事

存活判定只看进程本身，不碰关系库、内存态、授权服务或任何外部依赖；进程正在关闭时返回 503 并给出关闭原因。**就绪判定才是真正的流量准入**，控制面的就绪判定有八类独立的不可用原因：收到终止信号、运维主动排空、冷启动尚未完成、授权模型对账未热、密钥工作器未启动或异常退出、内环能力快照对账未成功、关系库不可用或仍在恢复、模型网关快照不可用（该项还覆盖"本进程缺少必要执行能力"与"共享热状态已降级但仍有策略需强制执行"两种情形）。具体是哪一类，通过响应头 `X-Lp-Not-Ready` 暴露。

> **说明**
> **为什么容器编排的健康检查故意只用存活判定？** 因为启动期的运行时漂移修复是一个后台控制环，需要时间收敛。若让编排器按就绪判定重启容器，收敛过程会被反复打断，表现为永远起不来的崩溃循环，而真实原因（某项对账还没跑完）永远不会显现。用存活判定做重启依据、用就绪判定做流量准入，是把"重启"和"摘流量"这两个动作分开。

启动宽限期按各依赖的真实冷启动时间设定，差异很大：主库 180 秒，模型网关数据面 600 秒且重试 30 次，护栏 30 次重试，其余平台服务 12 次重试。模型网关数据面的探针是复合的——先确认 REST 端口可连，再请求就绪端点，若声明了 gRPC 监听地址还要确认 gRPC 端口可连；只探一个端口会把"gRPC 监听丢失"误判为健康。

6.2 启动顺序里的四个刻意设计

前置校验先于一切。 运行代际校验任务是整个栈里第一个跑完的东西，迁移器、模型网关数据面库初始化与密钥服务迁移都要等它成功完成才开始（见 5.5）。把一致性检查放在最前面，是为了让"配置投影过期"这类问题在一秒钟内报出来，而不是变成三个服务轮流崩溃。

入口不声明任何依赖。 它先起来，后端未就绪时静态站点资源仍然可以送达，API 请求返回上游错误。如果入口等后端健康才起，一次后端故障会让客户连页面都打不开，只能看到连接被拒绝。维护态则由一份独立的入口配置承担：整站返回 503 并带 30 秒的重试提示、渲染静态维护页，同时保留内部接口、Webhook 与 OAuth 回调三类路径继续可达。

模型网关数据面自己有一段前置序列。 它先等待密钥服务下发的工作负载令牌文件出现，再执行自己的数据库迁移并激活写者，然后校验"确有活跃写者"，最后才启动服务进程。这四步是串行的，任何一步失败都不会让服务进程带着半套 schema 起来。

运行时数据面前置只等控制面"已启动"，而不是"已健康"。 理由写在编排注释里：控制面不可达时，它仍要靠已验签的磁盘快照先行服务。配套机制是配置分发的租约期 30 天、重铸周期 10 天，而健康快照的最大生命期只有 120 秒——控制面短时间不可达时数据面照常工作，但它对上游健康状态的认知会在两分钟内过期。

另有一处不对称值得注意：控制面对护栏的健康依赖标记为"非必需"，但护栏本身是硬依赖，未配置护栏地址时控制面直接拒绝启动。前者是启动时序上的宽容，后者是配置完整性上的严格。

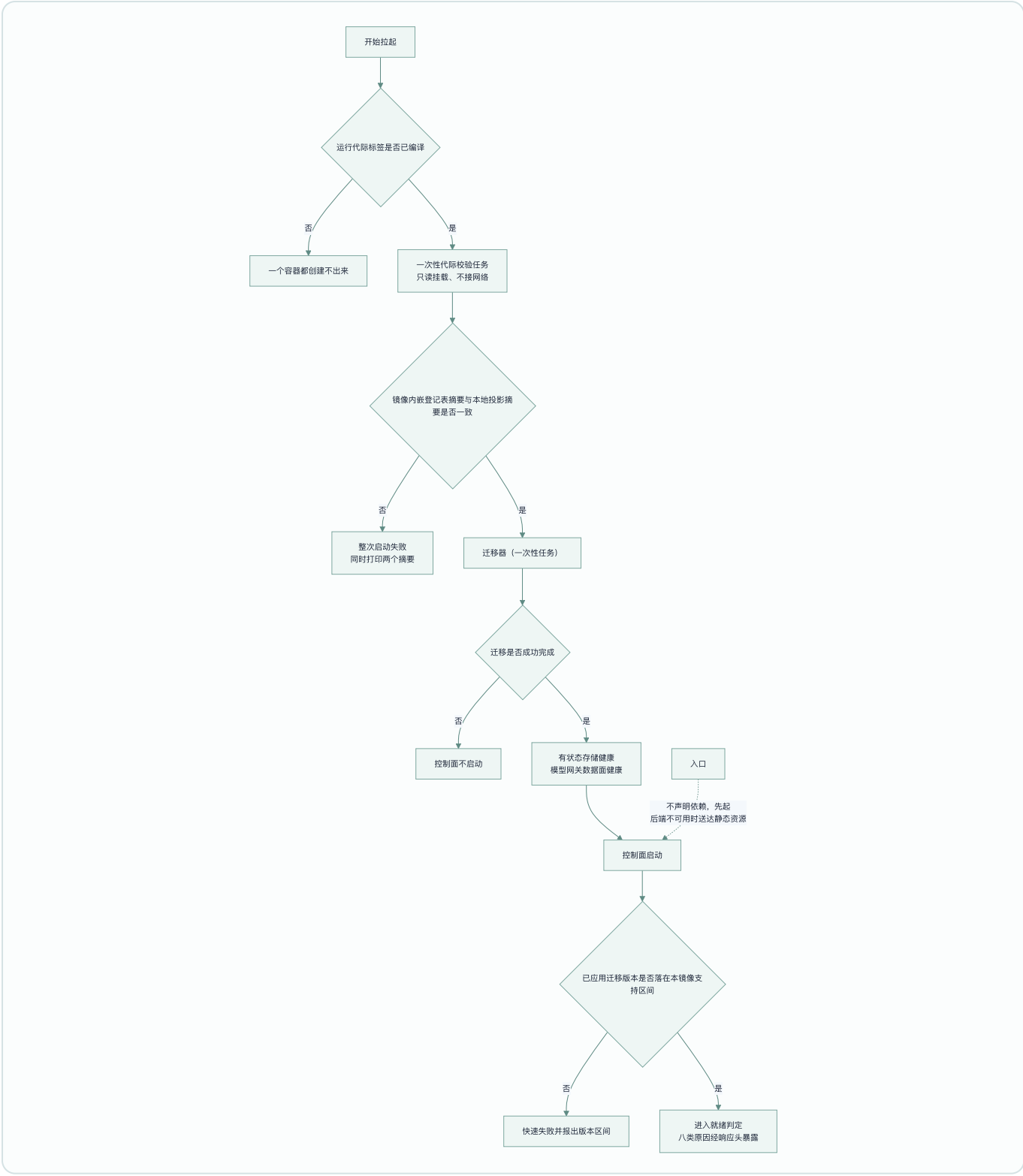


图 6–1　启动闸与依赖顺序

图注：这张图回答"栈起不来时该从哪一层开始查"——越靠上的闸失败得越早、报错越明确，最难排查的从来不是这些闸，而是没有这些闸时的表现。

6.3 故障域与降级矩阵

依赖	不可用时的影响面	姿态
授权服务	全部需要鉴权的请求	**拒绝**：租户闸与平台闸一律返回 503，从不放行；唯一读侧降级是浏览类查询的"最后已知良好快照"，只服务肯定结果，冷条目仍隐藏
内存态	限流与流量保护	**平台侧拒绝**：淘汰策略不合规时在请求期返回 503。**网关侧放行**：未配置流量内存态地址时限流闸失效，属刻意设计
护栏	受治理调用与内容检查链路	**拒绝**：未配置即拒绝启动；受治理调用执行面到外部工具的请求与响应链路失败即拒绝
模型网关数据面	全部模型调用	**拒绝**：无可用候选时返回 503 并给出无健康目标的错误码
密钥服务与令牌刷新器	模型网关数据面与平台侧的工作负载身份	**降级后拒绝**：令牌有效期 120 秒、每 30 秒刷新一次，刷新链路中断后约两分钟内令牌过期
控制面	管理面与策略变更	**降级**：模型网关数据面按已下发的策略继续服务，运行时数据面前置按已验签快照继续服务；上游健康认知在 120 秒后过期
对象存储	文件上传下载与审计归档	**降级**：控制面启动不阻塞（仅要求其已启动），其余业务路径不受影响
列式库	平台日志、观测事件与原始调用记录	**降级**：用量与成本真相在关系库投影中，计费不依赖它；但它是控制面的启动期健康依赖
客户自有监控体系	指标采集与看板	**无影响**：不在任何服务的启动依赖里

表 6–1　故障域与降级矩阵

6.4 优雅停机与它的边界

进程同时监听中断信号与终止信号；收到任一信号后，先把存活状态标记为"正在关闭"（此后存活端点返回 503），再广播停机，由 HTTP 服务器按优雅停机流程排空在途请求。信号处理器安装失败时，进程按停机路径处理而不是继续运行——这是一处失败即停的设计。

> **注意**
> **停机宽限与长响应的关系**：当前基线的编排文件中没有任何服务声明停机宽限期，实际宽限按容器引擎默认值 10 秒执行。与之对照的是模型调用的四段传输超时默认值——建连 10 秒、响应头 30 秒、流空闲 60 秒、单次尝试总时长 300 秒（数据库对这四个值分别强制 1–120 秒、1–300 秒、1–600 秒、1–1800 秒的取值范围并校验其先后顺序）。也就是说，**一次正在传输的流式响应最长可以跑 300 秒，而停机宽限只有 10 秒**：滚动重启会强制中断在途的长响应。需要平滑重启的场景，正确做法是在入口层面先摘流量、等在途响应自然排空，再停容器。

6.5 主库的 Zentrix 专属启动参数：含义与调整边界

本形态给主库下了一组专属启动参数，共六项，它们既是容量话题也是安全话题：

- **共享缓冲 256 MiB、有效缓存估计 512 MiB**：前者是数据库自己管理的页缓存，后者是它对"操作系统还会帮我缓存多少"的估计，只影响执行计划的选择。这两个值按"与其他容器共处一台机器"的前提取的保守值。宿主内存充裕且

主库是主要负载时，两者可以按比例上调；上调共享缓冲需要同步确认容器共享内存仍然够用。

- **单次排序内存 8 MiB**：这是**每个排序或哈希操作**的上限，不是每连接、更不是全局。峰值内存约等于这个值乘以并发中带排序的查询数，所以它是最容易被调大调出事的一项——调大之前先确认并发上限。
- **最大连接数 500**：这是连接数上限，不是并发处理能力。连接池对主库的后端连接上限是 48，通配库是 5，直连主库的是控制面、迁移器、护栏与模型网关数据面四类进程，因此 500 在默认拓扑下有充足余量；它的意义在于把"连接泄漏"挡在数据库崩溃之前。
- **口令加密为 SCRAM-SHA-256**：这一项不应改。
- **预写日志级别为 replica**：这是 PostgreSQL 的默认级别，决定预写日志里记多少信息。本形态的数据保护走 6.6 的业务备份集（主库用逻辑导出），这一项不需要按备份口径去调。

6.6 数据保护：备份集、恢复顺序与不存在证明

本形态的数据保护手段是**产品化的业务备份集**。它按平面组织：关系主库、模型网关数据面库、授权服务库、对象存储、网关台账、列式库、部署契约与配置指纹，另有三个状态平面（维护窗口、控制面写闸、写冻结）以标记形式写进备份清单。主库使用逻辑导出，恢复以整份备份为单位；配置平面只产出存在性指纹，不导出明文。

这里有一处专门为"多交付形态"设计的机制值得单独讲：**当前形态未配置的平面不会被静默跳过，而是必须留下"不存在证明"**。以向量索引平面为例，脚本发现该服务未在编排中配置时，会写下一条带独立证据 schema 的记录，明确标注"该服务未在本次部署中配置"，并把该平面判为已验证——而不是把它当作一次失败，也不是当作什么都没发生。恢复侧同理：条件平面只有在状态为"已验证"或"未配置"时才允许清理目标端残留，否则恢复直接终止并保留现场。这防的是一类隐蔽的数据错位——源端与目标端形态不一致时，静默跳过会让恢复后的环境带着上一次部署的残留数据继续运行。

备份清单还记录四项：已应用的迁移版本、应用版本、源环境标识与主密钥指纹。前两项决定这份备份能恢复到哪个版本的代码上，后两项决定它能恢复到哪个环境——主密钥指纹不匹配意味着所有派生凭据都对不上。

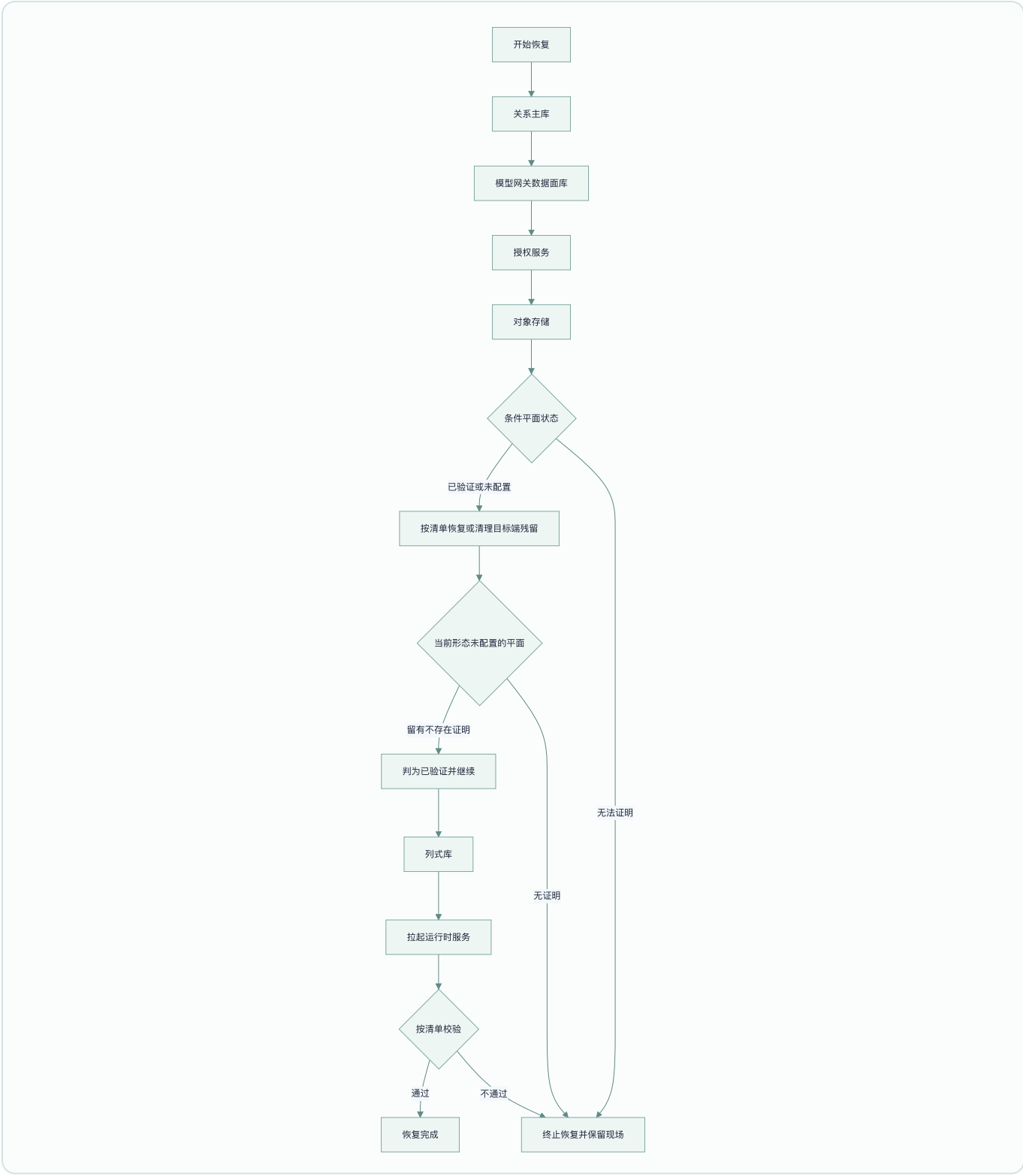


图 6-2　备份集平面与恢复顺序

图注：这张图回答"恢复到一半失败会停在哪里"——判据不通过时恢复终止并保留现场，而不是带着不确定状态继续往下走。

能力状态

业务备份集的产出入口是**宿主上的备份脚本**：跑一次产出一份带清单的备份集，**周期性执行由客户自己的定时任务承担**——包内没有产出本地备份集的定时单元，这一条必须在交付时落实到责任人。产品化的备份保护策略表记录的是口径而不是执行者：启用位默认关闭，计划字段默认每日 02:00、本地与异地各保留 3 份或 7 天，就绪状态初始为"未就绪"。异地拷贝定时器随包安装、默认禁用，通过 `LP_BACKUP_RESTORE_OFFHOST_SCHEDULE_ENABLED` 启用，且要求同时具备有效计划、平台密钥与计划令牌——三者齐备才会真正调度，避免"以为在拷、其实没有凭据"。恢复演练单元同样随包安装、保持禁用，演练由经平台接口创建的任务执行，因此每一次演练都留在任务历史里并受恢复门禁约束。

RPO 等于自建备份调度的周期。 把备份脚本挂进客户自己的定时任务、按每日一次执行，RPO 的上界就是一天；缩短周期可线性缩短 RPO，代价是每次逻辑导出对主库的读放大。**调度挂上之前，这套环境不存在恢复点承诺**——交付验收时应把"备份调度已落实并成功产出过一份备份集、异地拷贝与恢复演练各验证过一次"写进运维职责清单，未落实时据实标注。**RTO 由恢复串行顺序中各平面耗时之和决定**，而不是最大值；实践中通常由主库逻辑导入与对象存储回传两项主导。影响这两个数字的四个变量必须逐个代入：数据体量、磁盘吞吐、是否异地、是否需要重建索引。

6.7 容量由哪些变量决定

本书不给"推荐多少核多少内存"这类数字，而是给出它们的产生方式：容量结论由每个交付包自带的标定产物在安装时按目标机型判定并打印（见 6.8）。 不可复现的数字对客户没有价值，对交付方是长期负债。本产品对容量的处理方式是给出**变量模型**与**标定方法**。

Zentrix 的容量输入模型是公开的一组变量与系数，售前测算与交付前置校验共用同一组公式。它要求先声明业务口径：**用户数、每日工作时长、人均每小时请求数**——三者相乘得到日请求量，除以工作秒数得到平均每秒请求数，再乘以峰值系数得到峰值每秒请求数。然后声明观测口径：**每请求产生多少条事件、每条事件多少字节、仓储放大系数**，三者与日请求量相乘得到每日原始数据量。最后声明留存口径：**热留存天数与审计留存天数、归档份数**。

由这些输入推导出的容量分五类，每一类都另有一个下限值兜底，取推导值与下限的较大者：

容量类别	由哪些变量决定
列式库热数据	每日原始数据量 × 热留存天数 × 仓储放大系数
审计留存与归档	每日原始数据量 × 审计留存天数 × 仓储放大系数 × 归档份数
关系库	用户数 × 每用户字节 + 日请求量 × 热留存天数 × 每请求字节，再乘放大系数
内存态	用户数 × 每用户字节 × 放大系数
对象存储	日请求量 × 每请求字节 × 对象留存天数 × 放大系数

表 6-2　容量推导中的变量依赖

有状态热容量是上述四类在线部分之和；备份容量是它乘以备份份数；物理容量还要再乘存储副本数并加上余量百分比。**表中出现的每一个系数都是模型输入的默认值，不是标定结论**——它们的作用是让"我按什么假设算的"这件事可以被写下来、被质疑、被替换成客户自己的观测值。

注意

该模型模板里的运行基线字段（空载 CPU、空载内存、指标采集状态、各组件副本数、各存储卷容量）随模板发出时**全部标记为"未采集"**，模板本身写明：在现场或实验室产出证据并合入之前保持未采集状态，**不得用旧的验证数字手工填充**。这条约束是本产品容量口径可信度的来源，读者也应当用同一把尺子衡量任何一份声称给出了容量结论的材料。

6.8 怎么自己标定

单机交付形态另有一套**标定框架**，它规定的是"数字怎么被产生出来"，而不是数字本身：

- **独立性要求**：必须完成恰好 10 次完全独立的干净安装运行。脚本不重试失败的运行，也不把一次运行内的时间序列采样当作独立样本；已存在的运行输出会让脚本直接终止并拒绝复用。另有 3 次筛查跑数用于前置判断。
- **每次运行的通过门槛**：干净重置、静默窗口与基线回归三项状态全部通过；两段旅程都判定通过；代表旅程不少于 3600 秒（60 分钟）；内存溢出计数为 0、内核因内存不足杀进程次数为 0、无持续交换；必须留下写入增量与任务时延两类证据。
- **Zentrix 的固定旅程定义**：最小旅程是"全新安装、核心服务健康、版本回读"；代表旅程是"最小旅程 + 连续运行 60 分钟"。旅程定义按交付形态各自固定，因此不同形态的容量结论不可互相引用。
- **余量口径**：内存门槛取最小旅程峰值 ×1.2 与代表旅程峰值 ×1.3，按 GiB 向上取整；口径在代码里写死为"工程余量，不是统计置信度"——这句话明确拒绝把 1.2 与 1.3 解释成某个置信区间。
- **可比性要求**：结果按七个维度落入矩阵格——操作系统标识、操作系统版本、架构、容器引擎主次版本、存储后端、文件系统类型、旅程版本。七项必须完全一致，两次标定才可比。
- **适用边界**（原文口径）：标定结论适用于此产物、此精确矩阵格与此固定旅程；客户自有的后台进程、业务负载、其他文件系统或未声明平台需要另做一次标定。

标定产物随每个交付包单独生成。安装器用包内产物的精确矩阵格把宿主判成三档：**不可安全安装**（在做出任何变更之前硬阻断）、**仅体验**（只允许最小旅程，告警后继续）、**可做正式验收**。判据就是上面那两个阈值：宿主内存低于最低内存下限即阻断，低于代表旅程阈值即降为仅体验。旧的固定内存与磁盘阈值常量已下线，并有反向断言防止被重新引入。

磁盘的上界可算。 安装阶段所需空间由包内清单精确算出并由向导打印：公式是写入量乘以 1.2 后向上取整，再加一份保留量，保留量取"2 GiB"与"磁盘总量的 5% 和写入量的 25% 两者中的较小者"之间的较大者。运行期观测数据有硬上界：平台日志 50 GB、列式库日志类 14 天、会话用量 90 天、容器日志单文件 50 MiB × 10 个。向导另有一份明确标注"不参与安装硬阻断"的生命周期建议：系统盘保留 30 GB、镜像导入 80 GB、运行期增长 100 GB、日志保留 20 GB、应急清理 20 GB，合计建议 250 GB。安装耗时的向导预估为 20–40 分钟，是解压、镜像装载、启动与验证四段之和。

> **说明**
> 以上都是确定量或推导框架，不是性能指标。**并发数、每秒请求数与响应时延这三类指标本书一律不给**——它们高度依赖上游模型的响应特性、调用方的请求形态与网络条件，无现场标定时给出的任何数字都不具备可复现性。需要这三类数字时，正确做法是用 6.7 的变量模型确定压测口径，再在目标机型上实测。

6.9 运行期巡检

有一条巡检链路被刻意设计成**不依赖平台服务**：小时级的存储水位巡检以系统级定时任务运行，默认启用且无条件启用，带 2 分钟抖动、错过时会补跑，结果以行式 JSON 落盘。退出码有明确语义：1 为警告、2 为严重、3 为阻断，任何非零都映射成系统级失败，便于挂接寻呼。磁盘写满往往正是平台自身降级的原因，此时若巡检要经平台接口上报，就恰好在最需要它的时刻失效。此外，安装完成后还会自动执行一次只读体检（默认启用，超时 300 秒；超时或返回非零只发出告警，不影响安装成功判定）。

这条链路的价值正在于它的独立性：磁盘水位这一类最常见的运维故障，即使平台自身已经降级，仍有一份不依赖平台服务的旁路记录可查，并可直接挂接客户的寻呼系统。

6.10 这对你意味着什么

如果你是平台运维负责人，这一章给你四件必须在上线前落实的事：把备份脚本挂进你自己的定时任务并验证一次异地拷贝与一次恢复演练、确认 10 秒的停机宽限对你的流式调用场景是否可接受（默认单次调用可跑 300 秒）、在目标机型上跑一次容量标定而不是引用任何外部数字、以及把各服务的就绪端点接进你自己的监控体系。如果你是技术决策者，

需要写进评审结论的是：本形态对可靠性的承诺方式是给出可验证的机制与边界而不是未标定的指标；恢复能力按平面组织且严格串行，当前形态未配置的平面必须留下不存在证明而不是被跳过；备份与恢复的口径以业务备份集为准，RPO 由客户自建的备份调度周期决定、RTO 由各平面串行耗时之和决定。

第 7 章 · 私有化交付与运维

私有化交付的成本不在"装一次要多久"，而在三件事上：这个包里到底有什么、改一次配置会不会把它改坏、升级失败还能退回到哪一步。本章按这三个问题写 Zentrix 的交付形态、配置代际事务模型与升级回滚边界，最后把日常运维需要什么人、什么时候需要人，逐条列清楚。

7.1 交付形态：一个离线包、31 个编排单元

默认启用的主线路径是离线自解压包：引导脚本按处理器架构拉取发布清单，校验包名与摘要后下载，自解压包展开出安装向导与安装器，全过程不要求目标机器具备编译能力或访问代码仓库。第二条路径是纯离线归档包，解包后直接运行同一个安装器，用于介质拷贝交付与现场排障。两条路径落到同一个安装器，行为一致。

包内是 31 个编排单元，按面划分如下。

面	编排单元
入口	入口反代与 TLS 终止、网关证书重载器
控制面	控制面服务、数据库迁移器、配置代际校验器、按需的管理工具容器
治理与执行面	能力平面、受治理调用执行面、运行时数据面前置（职责见表 7–2）、护栏服务与其两个初始化单元、告警执行器、通知投递适配、计量执行器、维护执行器
模型网关	模型网关数据面、其数据库初始化单元
密钥服务	密钥服务、其迁移单元、两个工作负载令牌续发单元
有状态存储	关系主库、连接池、内存态、授权服务及其独立库与迁移单元、对象存储及其初始化单元、列式库

表 7–1　Zentrix 交付包的 31 个编排单元

其中 8 个是一次性初始化与校验容器（授权服务迁移、对象存储初始化、配置代际校验、控制面数据库迁移、护栏种子与护栏模型初始化、网关库初始化、密钥服务迁移），1 个是档位门控、按需拉起的管理工具容器，其余 22 个常驻。

其中有两组名称语序相近、容易读混（模型网关的数据面进程与平台侧的两个"网关"进程），本书统一如下，后文不再重复括注。

本书写法	进程名	本形态下的职责
模型网关数据面	zstack-router	承载全部对外模型调用：准入、路由策略、插件链、供应商适配、上游调用与用量记账
受治理调用执行面	lp-agent-gateway	受治理的 MCP 与连接器调用的单次执行：校验调用身份、在本进程内解密凭据、经护栏后派发
能力平面	lp-capability-plane	有效能力投影与绑定治理，直读关系库与授权服务，不回调控制面做鉴权
运行时数据面前置	lp-gateway	持有控制面签名下发的策略快照与本地台账，随包部署并常驻运行；另有模型出口与控制台访问前置鉴权两组路由，均以智能体身份为前提

表 7–2　四个进程侧术语的统一写法

这份包用一份替换版的入口站点配置，并对关系主库使用本形态专属的启动参数：共享缓冲 256 MiB、有效缓存估计 512 MiB、单次排序工作内存 8 MiB、最大连接数 500、口令加密算法 scram-sha-256、预写日志级别 replica。

三个独立执行器的存在有明确理由，包配置里逐条写明了：用量投影由**计量执行器**执行（控制面侧的同名开关保持关闭，保证同一份工作只有一个执行者），它是用量数据的唯一生产者；告警规则的**评估循环由告警执行器**承担，控制面只保留告警的管理接口；会话过期与留存清理由**维护执行器**承担。**交付包是按"哪条链路由谁执行"组织的，不是按目录组织的。**

安装前有一道只读预检，判定分五类：可安装、带告警可安装、需确认修复、不支持、检查失败。硬性阻断项包括容器引擎版本不低于 20.10、容器存储后端可证明、必需端口 80 / 443 / 3000 可用；amd64 原生包另要求处理器暴露 x86-64-v2 指令集。

> **说明**
> 操作系统支持矩阵分两档：Ubuntu 22.04、Ubuntu 24.04、Debian 12、Rocky Linux 9.7、RHEL 9.7 为已验证档；麒麟、openKylin、统信 UOS 为实验档。宿主前置依赖由目标机器提供：容器引擎与容器编排 V2 需在安装前就位（预检会逐项校验），因此交付包本身只带镜像与安装器。安装向导给出的耗时估计为 20 至 40 分钟，另一处文案写 30 至 40 分钟，两者都是估计值。

7.2 介质完整性链：五次校验各覆盖一段信任边界

从公网获取到容器起来，介质摘要一共被验证五次，每一次覆盖的是一段不同的信任边界。

校验点	覆盖的边界
发布清单中的包名、下载地址与摘要	传输通道或下载地址被替换
下载完成后的摘要，以及复制到受限临时目录后的二次摘要	落盘后被本机其它进程改写
自解压包对内嵌载荷的摘要校验	包体被截断或拼接
展开后对内层归档的摘要校验	内层内容与外层不一致
镜像归档校验和与分片装载检查点	镜像层损坏，或装载中断后续跑到了另一个包

表 7-3　介质完整性链的五个校验点

分片装载检查点绑定到当前包自身的摘要：换包再跑，旧检查点自动失效，不会出现"用一个包的进度续跑另一个包"的静默错配。下载地址另有白名单与路径穿越防护。

> **注意**
> 纯离线路径下，若介质中缺少与归档同名的校验和附件，向导会跳过该步并只打印告警。用离线介质交付时，校验和附件必须与归档一同拷贝。

7.3 配置代际事务模型：为什么配置要被编译成一代

这是整套交付体系里最值得单独讲的工程资产。传统做法是"一堆环境变量文件互相覆盖"，其失效方式是隐蔽的：改错一处，症状出现在三个服务之后，而且报错内容与根因无关。Zentrix 把部署配置换成了一个可编译、可校验、可原子切换的对象。

配置只有四个来源：运维手写的少量项、由主密钥派生出的全部集群密钥、由公网地址推导出的拓扑项、来自签名发布清单的包身份项。四者被编译成**一代不可变的运行时投影**并按服务切分——每个服务只能读到属于自己的那一份——然后原子切换"当前代"指针，同时保留上一代作为回滚基线。

强制力来自三处，都是 fail-closed：

1. 编排文件顶部用三个代际标签声明依赖，取值为空时容器编排直接报错，**一个容器都创建不出来**；
2. 每个服务的投影文件被声明为必需，缺失即启动失败；
3. 一个一次性前置校验容器在任何服务接触数据库之前运行，比对镜像内嵌的配置登记表摘要与本地投影记录的摘要，不一致就中止启动并同时打印两个摘要。它挂载只读、不接任何网络，因为它只做一次本地摘要比对。

第三道闸在 Zentrix、POC 与完整形态三个包中都存在，不是完整形态专属。

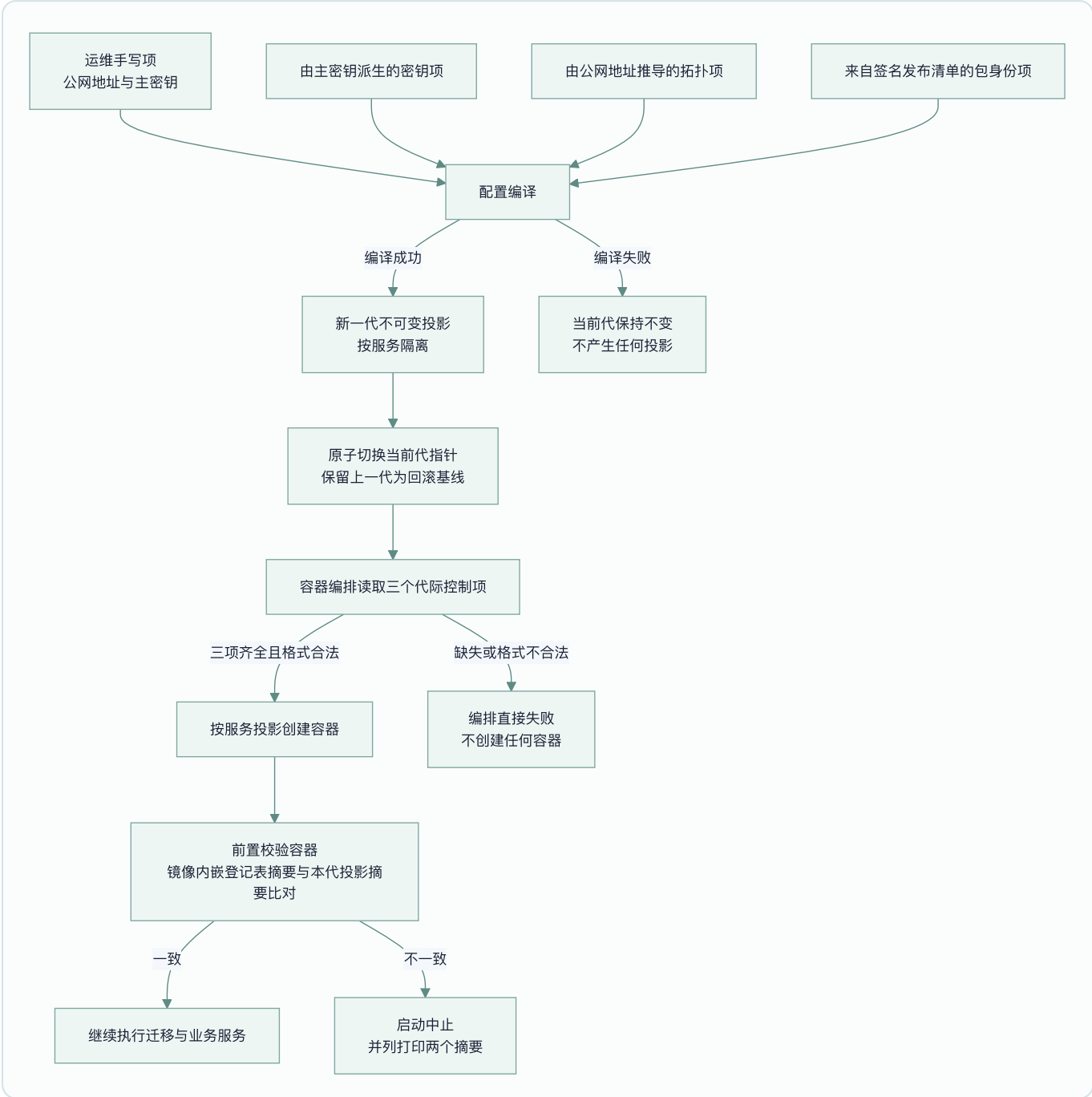


图 7–1　四源一编译与代际原子切换

图 7–1 回答的问题是：一次配置变更从哪里进入、在哪一步会被拦住、拦住时系统处于什么状态。三条失败分支的共同点是——都停在"旧代仍然生效"或"完全没有启动"，没有中间态。

这个模型消灭了三类事故。第一类是**手改派生文件造成的配置漂移**：派生文件带有"由生成器写入、请勿编辑"的文件头，每次运行整份重写，手改必被覆盖；正确改法是改源头再触发重新编译。第二类是**用旧投影喂新二进制**：新版镜像要求的键在旧代里根本没有被编译进去，症状是某个服务反复崩溃并抱怨"某个明明已经声明了的变量缺失"，根因完全指不到——前置校验闸把这个症状换成了一句话和两个摘要。第三类是**同一口径在多处重复定义**导致的分裂。

7.4 配置变更的事务语义

改一项配置不是"改文件再重启"，而是一次带预览、带锁、带备份、带验证、失败自动回滚的事务。

预览先打印四件事：将要设置的键与值（敏感值显示为已遮蔽）、代际标识从哪一代切到哪一代、**将被重建的且当前正在运行的服务清单**、以及随后要验证的五层。

应用先取排他锁，已有事务持锁时明确报错而不是并发写入；暂存改写结果后重新编译，若重新编译得到的代际标识与预览时不同，事务中止并要求重跑。随后备份运维文件（只保留最近两份）、原子替换、发布新代，再对受影响且正在运行的服务做强制重建。

验证分五层依次进行：源文件、投影文件、编排生效值、容器运行时、语义探针。任一层不通过即自动回滚到上一代并重新验证；回滚本身再失败，则写入阻塞记录并要求按指定代际显式恢复，而不是继续重试。事务默认超时 180 秒。

两条与凭据有关的硬规则：敏感键绝不接受位置参数或环境变量传入，只能从标准输入或无回显终端读取一次；一次性动作类的值（例如初始管理口令）在每次写入运维文件时被主动剥离，不会沉淀在配置里。

所有权模型决定谁能改什么：全部可写键共 101 个、归属运维；派生、拓扑、界面运行时、常量、包身份、一次性动作这六类所有权的键一律拒绝写入并提示正确改法。公网地址属于拓扑所有权，不能被直接改写，只能改源头再触发重新编译。登记表由三个交付形态共用，判断某个开关在当前部署里是否可调，先确认交付形态即可。

7.5 升级：整包升级这一条路径

Zentrix 的升级路径是**整包升级**：把同一交付形态的新版介质放到机器上执行，安装器自行判断这是新装还是升级，全过程用同一个安装器、同一套校验。这条路径的好处是升级前后的服务集合、镜像闭包与配置代际都可以逐项比对，不存在"某个组件被单独换掉、其余不知情"的中间状态。

> **高危**
>
> 跨形态介质被硬阻断，且刻意停在"解包之后、任何变更之前"：先按介质名预警，解包后再用包内清单做权威校验，已安装形态与目标包形态不一致时直接终止，并写下一份证据文件，同时明确告知业务数据与命名卷未被修改。因此 **Zentrix 与 Zenova 之间的形态变更必须按"换包部署 + 数据迁移"排期与验收，不能按"升级"估算窗口。**

升级序列有几个刻意的顺序安排：镜像先装载并校验，早于任何数据库迁移与服务变更；随后建立轻量恢复点（不复制数据库与镜像归档）；再重新派生密钥、重新推导拓扑并重新编译配置，之后才拉起服务。数据库迁移由编排中的一次性迁移容器完成，位于前置校验容器成功之后。健康等待默认 300 秒、另有 30 秒宽限、每 2 秒探测一次；通过之后仍会强制校验全部一次性容器的退出码、连接池连通性、控制面主密钥契约与网关令牌契约，最后比对运行版本与包版本。同版本重跑且健康时直接空转退出。

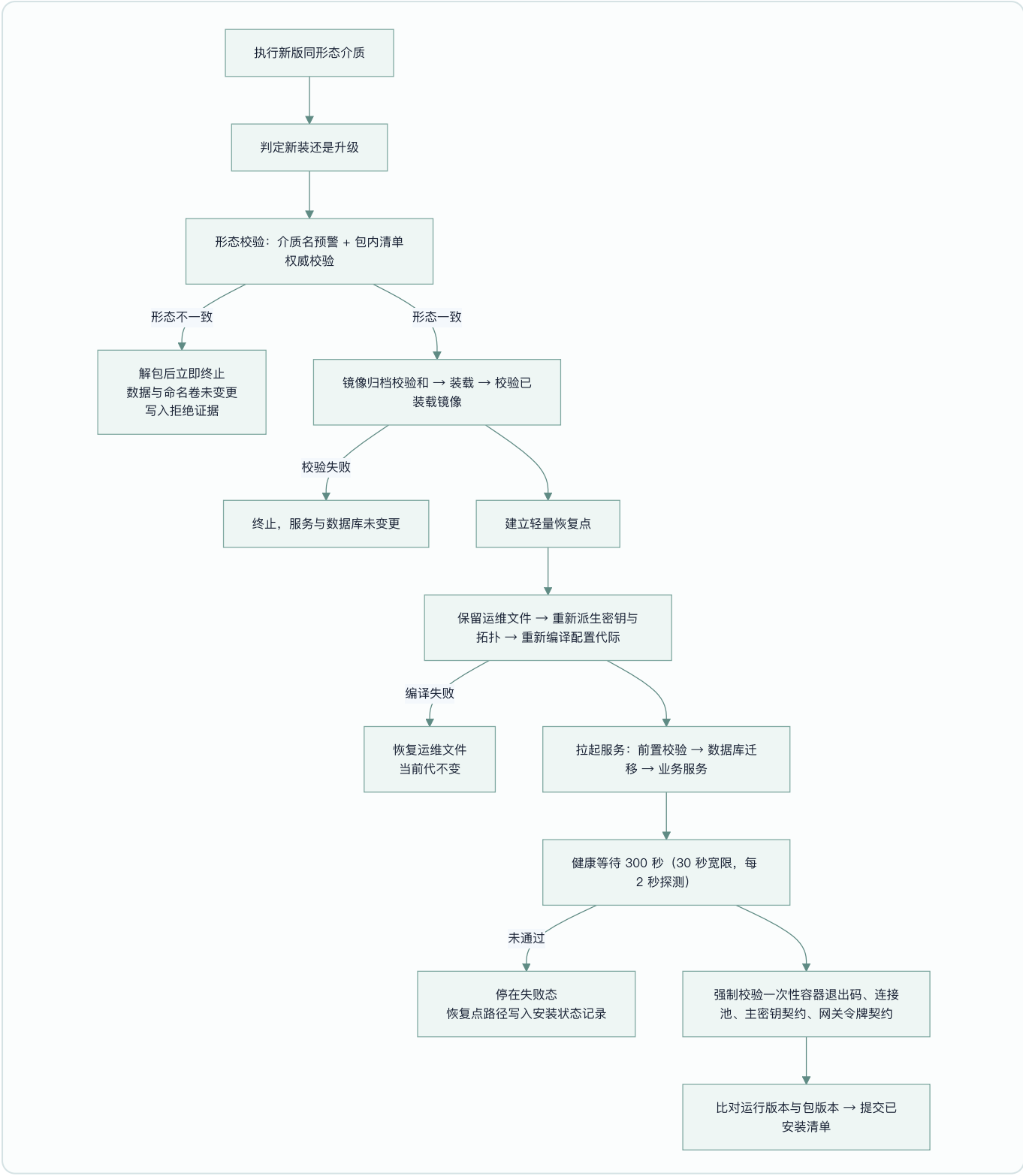


图 7–2　整包升级的阶段序列与失败落点

图 7–2 回答的问题是：升级在哪一步失败、失败时数据库处于什么状态。请注意分支 X0 与 X1 的位置——形态校验与镜像校验都在数据库被触碰之前完成，这两类失败不需要恢复动作。

高危

配置回滚不等于版本回退。产品提供的配置恢复动作只恢复配置文件、编排文件、包清单与运行时配置文件，随后停栈再起栈，**数据库内容与结构原地保留、不做任何变更**。真正的版本回退必须走恢复式路径：停栈、恢复升级前的数据库备份、把版本钉回旧标签、再起栈。原因是数据库迁移是单向的，旧版二进制不保证能在新结构上运行。任何把"回滚"理解成"一键退版本"的运维预案都需要重写。

7.6 日常运维：需要什么人、什么时候需要人

运维需要理解的配置面很小：真正必填的只有两项——公网基址与主密钥；可写键共 101 个，其余全部派生或有默认值。人员能力要求是 Linux 与容器基础，因为交付物是预编译镜像加单机容器编排，安装全过程没有编译步骤；不要求 Rust，也不要求 Kubernetes。

开机自启由一个服务单元承担。随包安装的定时任务有三个：每小时的存储水位巡检默认启用，且刻意不依赖平台服务，平台降级时仍能持续记录磁盘水位；备份恢复演练与异地备份拷贝两个单元随包安装但默认禁用。**产出本地备份集的周期性调度不在这三个单元里**——它由客户自己的定时任务承担，交付时必须落实到责任人。

已自动化（无需人工触发）	必须人工决策（附触发条件）
开机自启与每小时存储水位巡检	主密钥的保管与备份——它是唯一信任根，交付当天起即为一等运维职责
安装后自动执行一次只读体检（默认超时 300 秒，可关闭）	备份调度的落实与异地校验——交付时把备份脚本挂进客户自建的定时任务，并验证一次
端口占用自动避让（自首选端口起递增，最多尝试 100 次）	网关平面证书的更换——单机形态下由运维在到期前更换（换证经证书卷热重载，不需重启入口）
网段冲突自动换池（六组候选网段依次尝试）	升级窗口与回退预案的决策——每次收到新版介质时
镜像分片装载断点续跑、升级历史与恢复证据落盘	配置变更的执行与确认——每次改配置时，事务本身自动，决策不自动
配置变更的备份、五层验证与失败自动回滚	实验档操作系统上的风险承担

表 7–4　自动化与人工职责划分

按触发条件归纳，日常态下这套系统不需要固定排班的例行操作：巡检、自启、端口与网段冲突、镜像装载中断都已自动处理。需要人的时刻只有四类——收到新版介质（升级窗口决策）、要改配置（执行一次配置事务并确认五层验证结果）、证书临近到期、以及备份调度的落实与定期异地校验。

> **能力状态**
> 备份集由宿主上的备份脚本跑一次产出一份，**周期由客户自建的定时任务决定**；产品化的备份策略表记录本地与异地的保留、计划与就绪口径，启用位默认关闭。异地拷贝定时器随包安装、默认禁用，通过 `LP_BACKUP_RESTORE_OFFHOST_SCHEDULE_ENABLED` 启用，并要求有效计划、平台密钥与计划令牌三者齐备才会真正调度；恢复演练单元同样随包安装、保持禁用，演练由经平台接口创建的任务执行，每次演练都留在任务历史里并受恢复门禁约束。这几项在交付时按客户的备份策略逐条确认并各验证一次。

7.7 这对你意味着什么

为 Zentrix 配运维需要的是一个懂 Linux 与容器的人，而不是一个团队。日常态没有例行操作；需要人的只有四类时刻：升级窗口、配置变更、证书更换、备份调度的落实与异地校验。三件必须写进运维预案的事实是：升级用同形态的新版整包，跨形态介质在解包后被硬阻断；配置回滚只恢复配置文件、不动数据库，退版本必须先恢复数据库备份；主密钥是唯一信任根，它的保管从交付第一天起就是人的职责，产品这一侧会在指纹变化时显式失败。

第 8 章 · 开放性与集成

一套私有化系统的开放性不该用"提供丰富 API"来描述，而应回答三个具体问题：我已有的系统怎么接进来、我的数据怎么拿出去、我能在哪些位置扩展。本章只写当前基线真实存在且可用的集成方式，每条都标注启用方式与适用范围。

8.1 模型调用的对外协议兼容面

模型网关数据面对外提供两套入口，认证体系不同：网关原生入口使用网关自身的调用方密钥体系，平台兼容入口使用平台的租户与工作空间密钥体系。两者都由同一个数据面进程承载，都兼容主流大模型接口形态。网关端点类型中可路由的共 10 项，当前基线上已实现并可对外调用的是 9 个调用端点与 1 个模型列表端点：对话补全、响应式对话、Anthropic 消息、文本补全、向量化、图像生成、语音合成、重排序、词元计数，以及从租户模型路由投影生成的模型列表。

网关原生入口的调用方密钥支持三种呈递形式：Bearer 头、基本认证（其中用户名必须与令牌标识常量时间相等）、以及不带任何前缀的裸密钥。凭据以哈希形式存储；网关原生入口**每次请求都直接查库、不做认证结果缓存**——禁用与吊销因此立即生效，代价是每请求多一次数据库往返。

调用方密钥携带的生命周期与约束维度包括：有效期、吊销时间与吊销原因、绑定的网关模型与模型服务、声明的客户端类型与名称、计费绑定关系；网关原生密钥另有来源 IP 白名单、模型白名单、总配额与日月限额、每调用方令牌速率上限与队列优先级。把这些维度绑在密钥上，是为了让成本归因与吊销粒度落到同一个对象上：出问题时可以只吊销一个客户端，对账时可以直接回答"这笔花费属于谁"。密钥的创建、轮换与所有权转移需要租户管理员权限，其中所有权转移由 `LP_GATEWAY_MODEL_KEY_OWNER_TRANSFER_ENABLED` 启用——把转移做成一个要显式打开的动作，是为了让"密钥换主"始终是一次有意为之的运维决定。

> **注意**
> 模型白名单是准入边界，在任何路由工作发生之前评估，不通过直接拒绝；每调用方令牌速率上限在密钥上开启并设定上限后生效，判定窗口是真正的一分钟滚动窗口。gRPC 入口平面在出货登记表里监听 `0.0.0.0:3090`，单机形态下**只在容器网内可达**（该端口不发布到宿主，入口也不为它做反代），因此它的适用场景是编排网络内的集成方，对外集成走 REST 面；gRPC 反射通过 `ZR_GRPC_REFLECTION` 按需在内网调试时启用。

上游方向的兼容性同样是集成的一部分：网关调用上游时信任操作系统的根证书库，因此**由客户私有 CA 签发证书的自建模型端点可以直接接入**，不需要在产品内单独维护信任库。自建端点通过通用供应商接入，可配置 4 种鉴权方案、仅接受 POST 方法，线路协议取自全书统一的 17 项封闭枚举；上限为配置文档 64 KiB、端点数 16、模型规则数 256、单次调用超时 300 秒。通用供应商的模型列表推断在配置满足特定形状时成立。

> **说明**
> 供应商解析要求供应商标识、精确版本、端点键与网关端点类型四个事实全部精确匹配，并逐条禁止按域名推断、按模型名推断、未知供应商按兼容协议兜底等做法。接入自建端点的代价是必须显式声明这四个事实；换来的是一次错误路由（发错供应商、算错成本、把数据送到错误的上游）不会因为"猜对了大部分"而发生。

8.2 管理接口与规格

平台控制面与模型网关数据面各自有一份机器可读的接口规格，产品自身的管理控制台就是按这份规格生成客户端的，集成方可以用同样的方式生成自己的客户端。

两条边界需要明确。其一，控制面的规格端点与交互式文档端点由 `LP_OPENAPI_PUBLIC` 启用；入口另对这两个路径返回 404，作为纵深防御的第二道——把接口规格做成显式开启项，是为了让"对外暴露自己的接口面"始终是一次有意为之的决定。其二，模型网关数据面自己的规格端点无此开关，集成方可以直接据它生成客户端。模型调用侧的集成不依赖专用 SDK：它兼容主流大模型接口形态，现有客户端可直接使用。

审计数据有独立的导出面：提供结构化、表格与文档三种导出格式，大批量导出走异步任务加下载的方式，导出接口位于需认证的管理面内。这是"审计留痕可以带出系统"的具体兑现方式。

8.3 MCP 接入与分发

MCP 支持的协议版本集为 2025-06-18 与 2025-11-25（Streamable HTTP 传输），最新版本为 2025-11-25；候选状态的更新版本被显式拒绝，而不是"尽力兼容"。入口层暴露标准的授权发现端点（受保护资源元数据与授权服务器元数据，遵循 RFC 9728 与 RFC 8414），外部 MCP 客户端可按标准流程完成授权发现。

分发这一侧是 Zentrix 形态下的主用法：平台可以把一个受治理的 MCP 服务导出成客户端连接描述，并单独签发携带令牌的接入凭据；令牌可列举、可吊销，签发与吊销都留痕。占位版描述与带令牌版描述走不同接口，带令牌版只能从令牌签发接口取得。

执行这一侧的边界是本节最重要的部分：外部 MCP 客户端并不直连上游，请求由控制面校验令牌、租户与工作空间可见性、租户级路由准入之后，交给受治理调用执行面单次执行；**凭据由受治理调用执行面自己从已验证的凭据租约中解密，绝不从请求体读取**。这条链路上的护栏是 fail-closed——护栏地址未配置时相关服务拒绝启动。链路上还有流量保护，被限流、被熔断、保护降级会以各自独立的结果码返回，而不是笼统的失败。

请求体上限按用途分档：能力执行与 MCP 路径为 4 MiB，HTTP 透传路径为 100 MiB。MCP 授权的刷新令牌有效期默认 30 天，可配置范围 1 至 90 天。

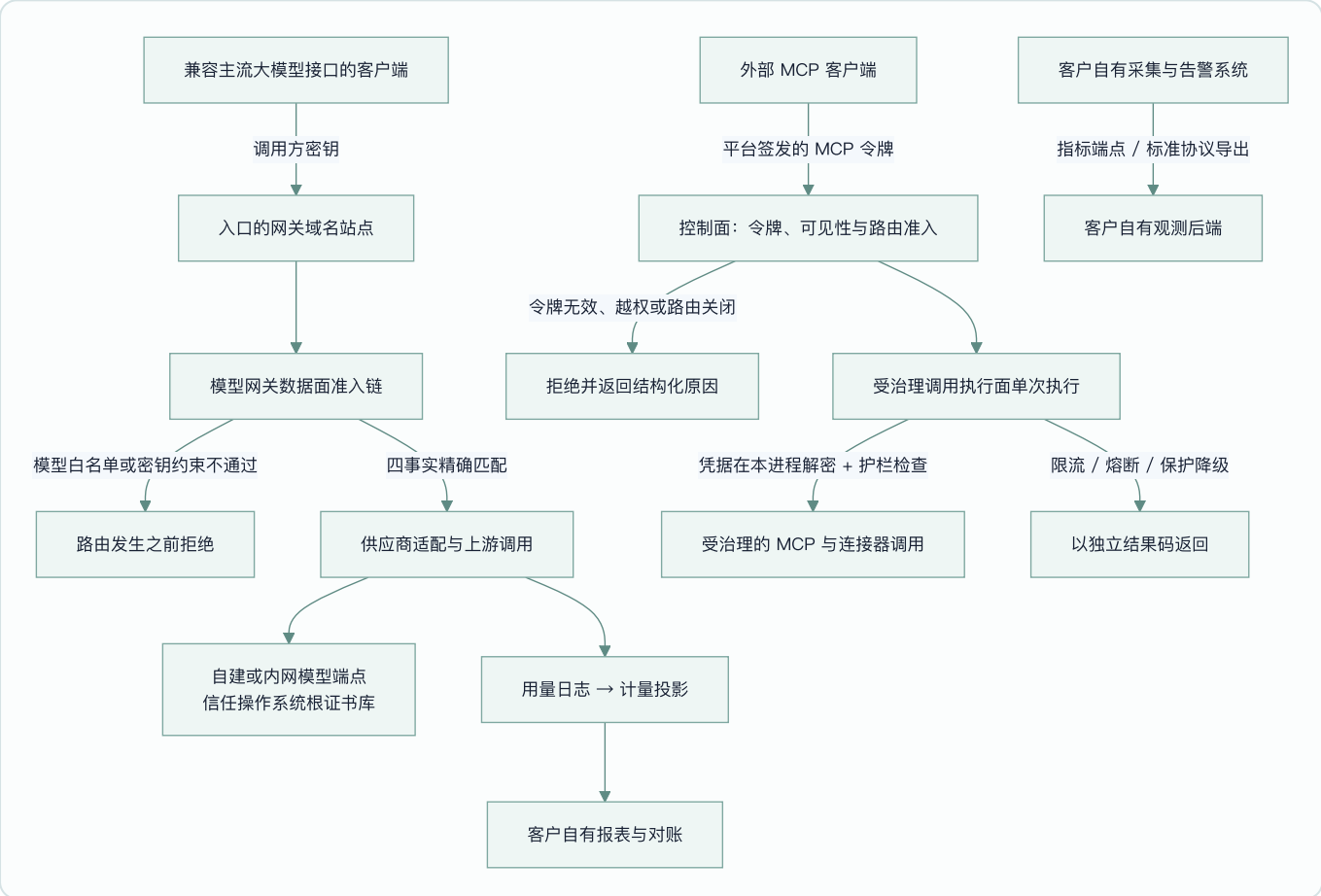


图 8-1　Zentrix 的集成面全景

图 8-1 回答的问题是：每条接入链路各自需要什么凭据、要过哪一道治理关口、被拒时停在哪里。两条主链路的共同特征是——拒绝发生在产生外部副作用之前。

8.4 两个扩展位：网关插件与统一路由策略

网关插件链是数据面上的扩展位，用 WASM 承载，有五个插入相位：模型改写（在模型准入之前）、请求门（路由选择之前）、路由感知门（路由选择之后）、请求变换（调用上游之前）、响应变换。硬限额为：燃料预算 200 万、线性内存 4 MiB、单次计算超时 20 毫秒、实例与表各 10。这个燃料预算比通用 WASM 沙箱的常见取值低两个数量级以上，理由写在代码里：数据面插件运行在热请求路径上，必须以微秒计而不是毫秒计完成。

插件平面随签名配置包下发。**插件在插件平面下发之后生效；一旦某条已发布的路由策略要求了某个插件绑定、插件平面却不可用，请求会被拒绝并返回明确的依赖等待原因，而不是跳过插件继续走**——这是刻意的 fail-closed。此外，插件的变换能力有硬边界：变换发生在模型准入、限流、路由选择与计价之后，因此它可以改写、脱敏或重塑内容，但不能把请求改到另一个模型或改变它的计费口径。

统一路由策略是第二个扩展位：12 类匹配条件（涵盖路径、请求头、客户端网段、分级、个人信息类型、模型、调用方、流量标签与调度窗口等），单值匹配与标签匹配使用两套故意不相交的运算符；动作包括加权目标选择、请求与响应变换、结构化变换、头改写、状态码映射、插件绑定与降级条件。目标选择策略有加权哈希与平滑加权轮询两种，**必填且无默认推断**。

> **注意**
> 响应体变换与降级判定的**适用范围是非流式的缓冲响应路径**，这是契约层写死的边界；依赖响应改写做协议适配的集成方案，需要把目标端点设计为非流式。此外，统一路由策略与网关策略集四族（请求阻断、防重放、词元上限、并发上限）均为策略驱动：由控制面发布对应策略后在数据面生效，因此上线前要把"发哪些策略"排进交付计划。

8.5 与既有身份体系的对接

企业既有的身份体系可以接进来做单点登录，当前有两条实现路径：通用 OAuth2 身份源，以及钉钉。两者的结构一致，配置按租户存放，客户端密钥加密存储。

两处启用动作需要明确：身份源按租户配置并打开启用位之后，这条登录路径即生效；用户即时开通（首次登录自动建号）另有独立启用位，打开时需同时配置好落地工作空间与角色，否则新账号没有可用的归属。回调流程中有一条硬约束值得写出来——**租户在服务端由登录态绑定确定，绝不从身份源返回的内容里推断**；账号状态类与配置状态类的失败一律收敛成同一个粗粒度提示返回给浏览器，精确原因只进审计，这是为了不把"这个账号是否存在、是否属于本租户"变成可枚举的信息。

8.6 与既有监控和数据体系的对接

"数据是你的"需要用具体出口兑现，Zentrix 形态下有三个出口。

用量与成本：网关自有的用量日志是一手真相，平台以固定 15 秒周期、每页 100 条的游标方式拉取并投影到关系库；产品侧的用量与成本口径读的是这些投影表，列式库已降级为原始观测用途。计量契约的金额以微单位整数表示且不得为负，调用结果为成功、失败、不确定三态——不确定态被单独保留，正是因为"上游是否已提交"在超时场景下确实无法断言。这条链路在 Zentrix 包里由独立计量执行器承担，是包配置刻意保留它的原因。

指标：模型网关数据面与各平台服务分别提供 Prometheus 文本格式的指标端点，可由客户自有采集系统抓取。这些端点或发布在宿主回环、或只在编排网络内可达，入口对指标路径一律返回 404，因此采集器需要部署在宿主上或加入编排网络，而不是从外部网络直接抓。指标遵守基数纪律，租户与工作空间标识明令不得作为指标标签；需要按租户归因时，走用量投影表与结构化业务事件，而不是给指标加标签——这是一条成本与可用性设计，它保证指标条数不随租户数线性膨胀。

关联标识：各服务的信号带统一的关联标识——请求标识、链路标识与遵循 W3C 规范的链路上下文头随请求贯穿传递，并写进结构化业务事件与容器日志。接入客户既有观测后端时，靠的是抓取上面这两条出口再按这组标识做关联，存放与展示由客户既有体系承担。

> **说明 · 本形态的信号出口只有上面三条**
>
> Zentrix 包在渲染阶段把 `OTEL_*` 与 `ZR_OTEL_*` 逐条移出各服务的配置面（发布配置的 `envDropKeys`），包内也不含采集器、时序库、链路库与看板。因此本形态的可观测出口就是指标端点、结构化业务事件与容器日志这三条，**模型调用正文不在其中**——链路标识只用于关联，不携带提示词与回复内容。

8.7 这对你意味着什么

把 Zentrix 接进既有架构时，四条路径是现成的：模型调用侧直接复用兼容主流大模型接口的客户端，凭据带有效期、吊销原因与计费绑定；MCP 侧可以把受治理的服务分发给外部客户端，凭据在执行面解密而不下发；身份侧可以对接既有 OAuth2 身份源；数据侧可以拿到用量投影、审计导出与指标端点。扩展位有两个，都在网关侧：WASM 插件链与统一路由策略。需要提前确认三件事：响应变换与降级判定适用于非流式路径；策略与插件由控制面发布后在数据面生效，上线前把"发哪些策略、装哪些插件"排进交付计划；可观测信号的接出按 8.6 那三条出口规划采集，看板与告警的二次编排落在客户既有体系。

第 9 章 · 技术选型说明与取舍

前八章说明了 Zentrix 是什么、怎么做的、每项能力怎么启用。本章回答最后一个问题：**为什么这么选**。

9.1 至 9.10 逐条按五段体例展开——面临的问题、备选方案、最终选择、放弃了什么、代价是什么；9.11 单独解释全书两种相反失败姿态的分岔依据，9.12 是十条决策的一览表。本章不收录只有优点的条目：如果一个选择的代价无法从当前交付基线的代码事实中读出来，那条决策就不写进本章。凡本章提到的数字与默认值，都取自 Zentrix 交付包的实际配置，而不是完整形态的配置。

9.1 为什么用 Rust 实现每一个服务进程

问题：私有化交付意味着运行环境不由我们决定。运行时越依赖外部装置（虚拟机、解释器、包管理器），交付面越宽，出问题时要排除的层次越多。而网关是全部模型流量的唯一通路，它多一层不可控依赖，整套部署就多一类不可解释的抖动。

备选：依赖运行时与垃圾回收器的托管语言；依赖解释器与运行期依赖树的脚本语言。

选择：两个相互独立的 Rust 工作区——平台侧 128 个成员单元，模型网关数据面 17 个 crate，后者在工作区层面把不安全代码设为禁止级别。每个服务以单一可执行文件进入自己的镜像，容器内不再需要语言运行时。类型系统让"失败即拒绝"成为可被编译器检查的性质：个人信息规则的动作字符串一旦无法识别，解析结果直接落到最严格的阻断而不是放行；签名配置包的载荷结构是唯一定义，上一版本的合法载荷在新版本下是反序列化失败而不是被按新语义误读。

放弃了什么：放弃了"改一行配置就能热更新逻辑"的灵活度，也放弃了大量现成的胶水生态。

代价：构建链本身成为工程负担。跨进程共享的契约结构要求同一处改动在多处同步落地——签名配置包的载荷结构必须同时改两个定义并同步两个派生函数，数据面回报结构必须同时改三处。少改一处不会在运行期被容忍，而是在反序列化阶段直接失败。

9.2 为什么把授权交给关系型授权服务，而不是在应用里写权限

问题：企业侧的授权不是"用户有没有某个角色"，而是"这个人通过哪条关系链拿到了这个对象的这个动作"。部门是树、职能组是并集、密钥与渠道的可见性是多轴叠加。用应用内的权限表表达这些，每加一条链路就要改一次查询，而每一次改查询都是一次可能放行越权的机会。

备选：在业务库中维护角色表与权限表，判定用连接查询完成。

选择：授权模型定义在仓库内的单一 schema 文件中，共 19 类对象，由控制面在启动时做差分迁移。组成员关系可递归嵌套，一次写入即让部门授权沿子树级联；等保 2.0 的三员互斥（配置管理员、安全管理员、审计员，另有运维员）可直接建模，而不是靠应用层判断。执行面判定强一致，进程内只缓存肯定结果（上限 10000 条、生存期 5 秒），浏览类查询另有一份只含肯定结果的长快照（上限 50000 条、生存期 600 秒）。

放弃了什么：放弃了"权限判定就是一次本地查询"的简单性，也放弃了在应用里临时加一条特例的能力。

代价：多了一个有状态依赖，且它自带独立的数据库实例与数据卷，备份恢复必须把它算作独立平面。它的可用性成为平台可用性的一部分——**不可达时一律拒绝而不是放行**；为避免抖动直接变成雪崩，客户端侧带熔断（错误率阈值 0.5、闭合窗口 50 次、半开 5 次、打开后等待 30 秒）。授权模型的演进也受约束：删除关系属破坏性操作，必须先列入白名单（当前只有 2 项），否则迁移拒绝执行。

9.3 为什么把配置编译成不可变代际

问题：运行时配置是"运维手写项 + 由主密钥派生的密钥 + 由公网地址推导的拓扑 + 包身份"四类来源的合成产物。如果容器直接读这份合成产物而没有任何东西校验它，那么一个容器读到几天前编译的投影、而它的二进制期待的是新登记

表里才有的键——运维看到的现象是服务反复重启、日志说"某某键必填"。这条消息完全指不到根因：那个键其实已经声明了，只是没有被编译进当前投影。

备选：容器直接读运维手写的环境文件；或者只在启动时打印一条告警，让运维自己判断。

选择：把配置编译成一代不可变的投影，并原子切换"当前代"指针；每个服务只读到属于自己的那一份。这道一致性在三个层次上都做成硬失败：

1. **编排层**：编排文件顶部的三个运行时代际标签使用缺值即报错的语法，未编译代际时一个容器都创建不出来。
2. **校验层**：一个一次性前置校验服务比对二进制内嵌登记表的摘要与已发布代际记录下的摘要，只读两个文件、只做一次比较，不一致即失败退出。
3. **依赖层**：数据库迁移器、模型网关的库初始化任务与密钥服务迁移器，三者都以"前置校验成功完成"为启动条件，校验没过就没有任何一次迁移会发生。

这三道闸在 Zentrix、POC 与完整三个交付包中都存在。

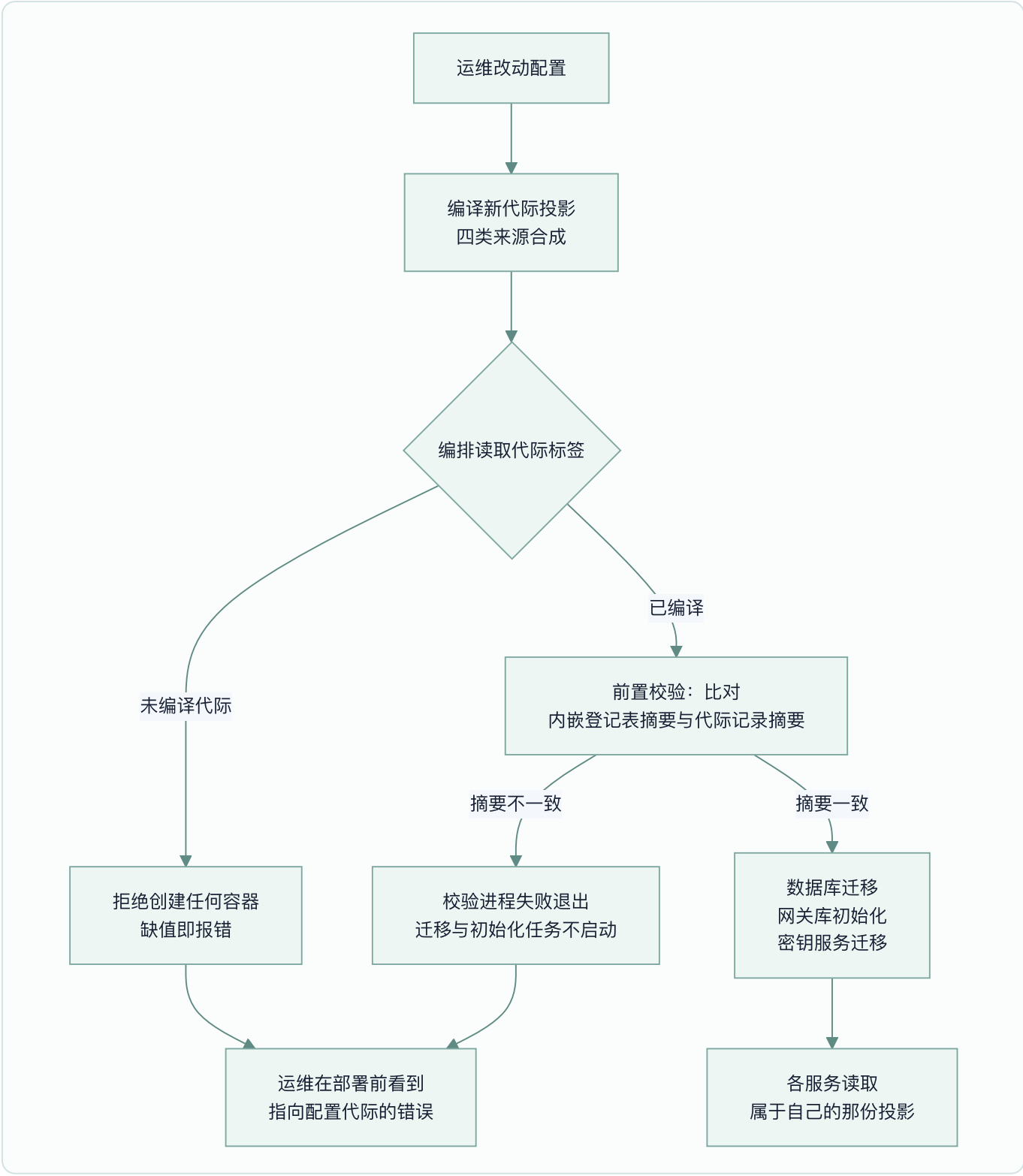


图 9–1　配置代际的三道闸与两条失败路径

这张图回答的问题是：为什么一次配置改动必须先编译再部署，以及配置陈旧会在哪一步、以什么形式被拦住。

放弃了什么：放弃了在机器上手改一个文件就让服务生效的能力，也放弃了"不同服务共享同一份环境文件"的便利。

代价：任何配置改动都必须走编译与代际切换，没有临时手改的路径。派生文件被手工编辑不会被静默接受，而是在下一次校验时表现为失败。交付包之间的配置面差异也由这一层产生：每个交付包的配置面在打包阶段按"本形态实际有消费者的键"渲染，因此同一个开关名在不同交付包里可能落在不同的配置面上——判断"这个开关在当前部署里能不能调"，先确认交付形态即可。

9.4 为什么默认交付是单机容器编排而不是只提供 Kubernetes

问题：客户机房的常态是"一台服务器、一次交付、一个运维"。把最小交付单元定成 Kubernetes，等于把编排系统的运维成本转嫁给客户，而这部分成本与网关本身要解决的问题无关。

备选：只提供 Helm 与 Kubernetes 交付，用编排复杂度换取弹性与高可用。

选择：默认交付形态是单机容器编排栈——Zentrix 包 31 个服务、14 个命名卷、3 张容器网络。规模差异用**交付形态**匹配，而不是用编排复杂度：同一份编排基线按服务白名单渲染出完整形态 52 个服务、POC 形态 34 个、Zentrix 形态 31 个。这样故障面可以逐个枚举，运维需要理解的对象是容器与卷。Zentrix 的关系主库还带一组本形态专属的启动参数——共享缓冲区 256 MiB、有效缓存估计 512 MiB、单次排序内存 8 MiB、最大连接数 500、口令加密算法固定为 SCRAM-SHA-256、预写日志级别为 replica——按本形态自己的连接与内存画像给定，而不是沿用其他形态的取值。

放弃了什么：放弃了把弹性伸缩与高可用做成默认能力。

代价：高可用必须另走 Helm 路径。Zentrix 有对应的 Kubernetes 部署模式与专属参数文件，该路径按现状标注——chart 版本 0.1.0，离线升级经镜像仓库策略交付。单机形态另有一处边界要写进运维预案：编排层未声明停机宽限，长连接排空受容器引擎默认宽限约束（见 6.4）。

9.5 为什么模型网关数据面与控制面分成两个进程

问题：模型流量的可用性不应被管理面的可用性绑架。若控制台故障导致全部模型调用失败，一次管理面滚动升级就是一次全域故障。而如果数据面每次调用都回控制面取配置与鉴权结论，那么两者在可用性上就是同一个进程，分不分开没有意义。

备选：数据面每次调用回控制面取配置与鉴权结论；或者把两者合成一个进程，用线程隔离。

选择：分成两个进程，并且切断运行期的同步依赖。模型网关数据面只等控制面"已启动"而不是"已健康"；入口不声明任何依赖，先起来，后端未就绪时静态站点资源仍可送达（Zentrix 的替换版入口配置不做后端错误的维护页回落，维护态由一份独立配置承担）。控制面向数据面下发的是**签名配置包**：Ed25519 签名覆盖整个载荷的规范编码，当前 schema 版本为 6 且没有兼容窗口——旧版本载荷在反序列化阶段直接失败，而不是被按新语义误读。配置租约 30 天、超过租约三分之一的复用需重铸并重新签名；健康事实的生命期只有 120 秒，且从编译时刻起算而不是应用时刻。路由投影租约与配置租约等长，但它编码的是"控制面仍可达且仍为这份投影背书"而不是内容新鲜度，因此一次通过认证的"内容未变更"应答同样续租——否则一个路由长期没改动的租户会在健康状态下被拒绝。

放弃了什么：放弃了"改一条配置立刻在数据面生效"的即时性，换成一条有版本、有签名、有租约的下发链路。

代价：配置一致性变成一整套需要维护的契约，而不是一次查询。载荷里有一个"未分配"标志位专门防止误删——配了选择器但本实例未被匹配时各平面为空且该位为真，应用方不得裁剪已有行。契约的增删字段要求发布方与应用方同步演进，任何一侧抢跑都会被显式拒绝而不是被容忍。

9.6 为什么模型解析禁止一切推断兜底

问题：一次错误的供应商解析意味着三件事同时发生——请求发到错误的厂商、成本记到错误的科目、企业数据流向未经审批的上游。这三件事的代价都远大于"少填一个配置项"。

备选：按域名推断供应商，按模型名推断线路协议，未知供应商默认按兼容形态调用。

选择：运行时解析必须四个事实全部精确匹配——供应商标识、精确的供应商有效版本、端点键、网关端点类型。渠道级协议类型、按基址域名推断、按模型名推断线路协议、未知供应商默认按兼容协议调用、别名解析——这五类兜底被逐条禁止，任一不一致必须在发出网络请求之前拒绝。供应商目录里的默认基址只是管理台的便利默认值，运行时不作为来源权威。当前内置 54 个供应商标识与 1 个可配置的通用供应商，全部是编译期常量；线路协议 17 种，其中 5 种支持流式。

放弃了什么：放弃了"先跑起来再说"的接入路径，也放弃了自动识别新厂商的能力。

代价：接入新供应商必须显式声明这四个事实。通用供应商虽可配置但带硬上限：配置载荷 64 KiB、最多 16 个端点、256 条模型规则、单次最长超时 300 秒。

9.7 为什么三套限流并存而不统一

问题："每分钟请求数"这个词在系统里出现于三处，语义并不相同。统一它们看起来是好事，但三者的作用域、时间语义与失败姿态本来就不一样，强行统一会迫使最关键的那一层接受错误的失败姿态。

备选：抽出一套统一限流内核，三处共用。

选择：保持三套独立实现并公开说明差异。其一，每调用方令牌的速率与词元限制走真正的 1 分钟滚动窗口，计数后端不可用时**拒绝**；其二，网关侧的通用速率限制基于共享内存态，周期是显式枚举的秒、分钟、小时，"每分钟一次"不会被折算成整数秒速率，命中动作有拒绝、排队、返回默认结果三种，冲突时按拒绝优先于排队、排队优先于默认结果裁决；其三，平台侧的 HTTP 中间件用秒桶，配置后端未就绪时**拒绝**。网关侧刻意选择在计数后端不可用时放行，理由是它是全部模型流量的唯一入口，一次抖动会同时让所有租户收到服务不可用。

放弃了什么：放弃了一份统一的限流语义文档，也放弃了"三处配置项可以互相换算"的便利。

代价：文档与运维认知负担。同一个字段名在两处语义相反，运维看控制台数字时必须知道自己在看哪一层。本书用第 3 章的对齐表（表 3–2）偿还这笔债；网关侧的放行姿态由一个独立的可观测计数器兜底，据此可以判断"是没有触发限流，还是限流本身没生效"。

9.8 为什么主密钥不可原地轮换

问题：整套部署只有一个信任根，34 种集群密钥与每个租户的数据加密密钥都由它派生。允许运行中更换它，等于允许信任根在运行中分叉——一部分数据用旧密钥加密、一部分用新密钥，而系统无法判断某一行属于哪一边。

备选：支持双密钥并存期，逐步重加密存量数据。

选择：不支持原地轮换，并把这个约束做成显式失败。Kubernetes 侧检测到既有配置属于不同信任根时直接终止；安装链路在主密钥指纹变化时告警说明全部派生密钥已轮换；护栏服务启动时用主密钥重新派生调用方密钥，与加载的钥匙串做常数时间比对，任一不符拒绝启动——这一条专门拦住"升级时跳过了配置步骤、于是带着开发密钥上线"。

放弃了什么：放弃了把主密钥当作可定期轮换的普通凭据来管理。

代价：主密钥保管成为一等运维职责，而不是可以事后补救的配置项。它同时是数据加密根，丢失意味着供应商凭据密文、能力绑定密文与脱敏原文存证都不可解。缓解手段只有两个：指纹校验（备份清单同时记录已应用的迁移版本与主密钥指纹，用来判断一份备份能恢复到哪个环境）与显式失败。

9.9 为什么把可观测信号与观测栈解耦

这是一个真实的取舍，正面回答。

问题：一整套可观测栈是十几个额外容器——采集器、容器套接字代理、指标、告警分发、链路、日志、看板，加上主机、容器、数据库、内存态与探活五类导出器。它们本身要占资源、要留存数据、要单独运维。而 Zentrix 的定位是"一台机器上跑起来的企业级 AI 网关"，把一整套观测系统绑进去，等于强迫每一个只想要网关的客户同时接受一套观测系统的运维成本，并且与客户已经在用的监控体系重复。链路数据在大模型场景下还多一层合规考量：链路属性里可能带着提示词与回复正文。

备选：随包交付全套观测栈，客户的既有体系放着不用；或者反过来，什么信号都不给，让客户自己想办法。

选择：两者都不选。Zentrix 产出标准化的信号，由客户既有的监控体系承接；同时随包保留三条不依赖任何观测栈的信号轨道：

- **列式库随包交付**，是日志、观测事件与原始用量的落点，用量与成本的投影表建立在它与关系库之上，因此计量与成本归因是一条自洽的闭环链路。

- **各服务自身的信号端点随包提供**，入口、控制面、模型网关数据面与各执行进程都在自己的端口上提供指标端点，客户既有的采集系统可以直接对接。入口在边缘对指标路径返回 404，是要求从内部采集而不是从公网采集。
- **告警执行进程与通知适配器随包交付**，业务规则告警与护栏命中台账到已绑定聊天渠道的投递轨道在 Zentrix 形态下是完整的。

放弃了什么：放弃了"看板随包即得"这一种交付方式，改为把信号标准化之后接进客户既有体系；也放弃了在网关产品里再维护一套指标存储与可视化的选项。

代价：客户需要把这些信号接进自己的采集系统，这是一次一次性的集成工作；接完之前，排障的起点是容器日志与各服务自身的信号端点。换来的是网关侧的资源占用与运维对象都收敛在 31 个服务之内。

> **说明**
>
> 解耦落到 Zentrix 包上是一次渲染期的裁剪：`OTEL_*` 与 `ZR_OTEL_*` 被逐条移出各服务的配置面，包内也不含采集器、时序库、链路库与看板。因此本形态接出去的是指标端点、结构化业务事件与容器日志三条，链路标识只用于把同一次请求在这三样信号之间对齐，**不携带提示词与回复正文**。

9.10 为什么容量口径用标定框架而不是推荐配置表

问题：售前最常被问"需要多少核多少内存"。给一个数字很容易，但没有标定产物背书的数字不可复现——换一个操作系统版本、换一个文件系统就不成立，而客户会拿它做采购决策。

备选：给出一张最低配置与推荐配置表，用经验值填充。

选择：把容量结论定义成一套标定框架的输出而不是常数。框架要求 10 次完全独立的干净安装运行，并拒绝把已存在的运行结果复用为独立证据；旅程定义按交付形态分别固定，Zentrix 形态的最小旅程是"全新安装、核心服务健康、版本回读"，代表性旅程是"连续运行不少于 3600 秒"；每次运行必须满足零内存溢出、无持续交换等门槛才计入；结果按 7 维矩阵格落格（操作系统、版本、架构、容器引擎主次版本、存储后端、文件系统、旅程版本）。内存门槛按最小旅程峰值的 1.2 倍与代表旅程峰值的 1.3 倍取整得出，口径写死为"工程余量而非统计置信度"，并附带适用边界：结论适用于此产物、此矩阵格与此固定旅程。

放弃了什么：放弃了在售前阶段给出一个可以写进方案书的配置数字。

代价：每个交付产物都要跑满标定才有结论。作为直接后果，早期那批固定的磁盘与内存阈值常量已从安装链路移除并有测试禁止其重新出现——标定产物缺席时，安装器只能给出分阶段的存储预算判定，不能回落到一个"看起来安全"的默认值。

9.11 失败姿态为什么不统一

本书反复出现两种相反的失败姿态。它们不是实现不一致，而是同一条原则在不同位置的不同结论：**当一层的判定关系到"是否越权、是否谎报"，它失败即拒绝；当一层是全部流量的唯一入口且判定只关系到"是否限速"，它失败即放行并留痕。**

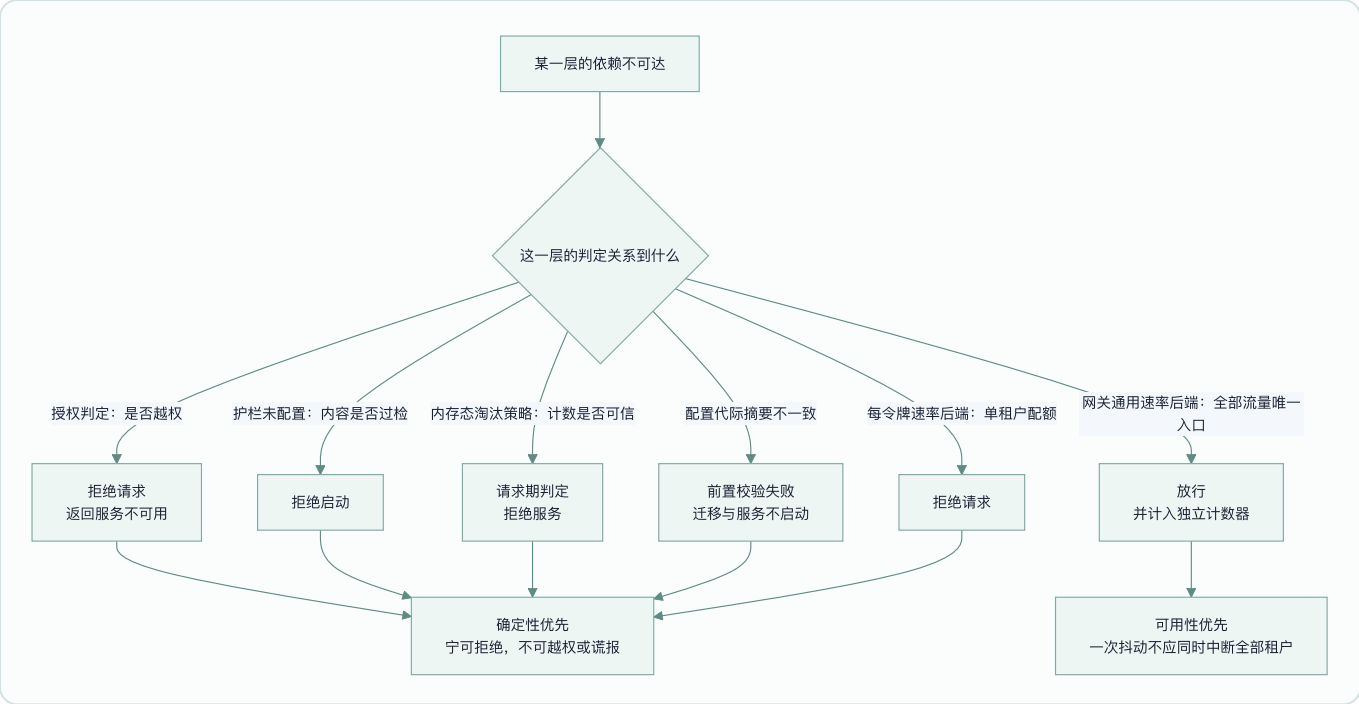


图 9–2　同一类故障在不同层上的失败姿态分岔

这张图回答的问题是：为什么"默认失败即拒绝"这句话不能一概而论，哪一处是刻意的例外，以及例外靠什么补偿。

9.12 选型决策一览

决策	备选方案	我们选了什么	代价
实现语言	托管运行时 / 脚本语言	两个 Rust 工作区（平台侧 128 个成员单元、模型网关数据面 17 个 crate），单可执行文件交付	跨进程契约改动须多处同步，少改一处在反序列化阶段直接失败
授权模型	业务库中的角色表与权限表	关系型授权服务，19 类对象，执行面强一致，只缓存肯定结果	多一个有状态依赖与独立备份平面；不可达即拒绝；删除关系属破坏性操作
配置模型	容器直读手写环境文件	编译成不可变代际并原子切换；编排、校验、依赖三层硬失败	不能手改文件；配置面因包而异，判断开关可调性须先确认交付形态
默认编排	只提供 Kubernetes 交付	单机容器栈（31 服务 / 14 卷），用交付形态匹配规模	高可用须另走 Helm（chart 0.1.0，离线升级经镜像仓库策略）；停机宽限取容器引擎默认值
面分离	数据面每次回控制面取结论	数据面只等控制面已启动；配置经签名下发，schema 无兼容窗口	需维护签名、租约、未分配位与回报的整套契约，两侧必须同步演进
供应商解析	按域名或模型名推断，未知按兼容调用	四事实精确匹配，五类兜底逐条禁止，发请求前失败即拒绝	接入新供应商须显式声明四个事实；通用供应商带硬上限
限流分层	抽出统一限流内核	三套独立实现，失败姿态按各自作用域分别选择	认知负担；同名字段两处语义相反；靠对齐表与独立计数器偿还
信任根	双密钥并存期，逐步重加密	单一主密钥，不可原地轮换，变更即显式失败	主密钥保管成为一等运维职责；丢失则多类密文不可解

决策	备选方案	我们选了什么	代价
观测边界	随包交付全套观测栈	产出标准化信号（列式库、各服务信号端点、业务告警与护栏命中投递），由客户既有观测体系承接	信号需一次性接进客户的采集系统；接完之前排障起点是日志与信号端点
容量口径	最低与推荐配置表	标定框架：10 次独立运行、7 维矩阵格、工程余量取整	每个产物须跑满标定；售前不能报数；旧的固定阈值常量已移除

表 9–1　选型决策一览

9.13 这对你意味着什么

如果你在做选型评审，本章的用法是逐条核对代价而不是逐条核对优点。这十条里有五条会直接落到运维流程上：主密钥保管、配置只能编译不能手改、限流语义的运维培训、供应商接入的配置纪律，以及"信号由包内产出、观察由既有监控体系承接"这一条对监控体系的接入要求。其中任何一项与既有流程冲突，都是选型阶段该谈清楚的事，而不是上线后再发现。

附录 A · 组件、端口与能力一览

本附录是全书数字的汇总，供评审与安全审查逐行勾选。口径与正文一致：端口写裸数字，网段写 CIDR，能力只有默认启用与可配置启用两种标注。**本附录只描述 Zentrix 交付包**。

Zentrix 包共 31 个服务：22 个常驻服务、8 个一次性初始化任务，以及 1 个仅在运维动作时启动的工具容器。命名卷 14 个。编排声明三张网络，本形态实际有成员的是两张。

A.1 常驻服务、端口与暴露方式

暴露方式分三档：**唯一对外**指在宿主全部网络接口上发布；**仅回环**指只发布到宿主回环地址；**不暴露**指只在容器网络内可达。

组件（进程名）	在 Zentrix 形态下的职责	端口	暴露
入口（Caddy）	唯一南北向入口、TLS 终止、反代；模型网关子域名直连模型网关数据面	80 / 443；管理 2019、指标 9091	唯一对外（管理与指标不暴露）
入口证书重载器	监视网关证书卷并触发入口重载，与入口共享网络命名空间	无独立监听端口	不暴露
控制面（lp-server）	平台 API、管理后端、受治理 MCP 的对外入口	3000	仅回环
模型网关数据面（zstack-router）	模型调用的准入、路由、协议变换与用量记账	3080 REST / 3090 gRPC	不暴露
能力平面（lp-capability-plane）	有效能力投影与绑定治理，直读数据库与授权服务	3020	仅回环
受治理调用执行面（lp-agent-gateway）	受治理的 MCP 与连接器单次调用执行，凭据在此跳解密；编排中标注为关键等级	3021	仅回环
运行时数据面前置（lp-gateway）	保持与控制面的快照刷新与本地台账；见本节末尾的说明	3010	仅回环
护栏（lp-guard-service）	内容安全与个人信息保护	3050 业务 / 3051 运维	不暴露
告警服务（lp-alert-service）	平台信号导出与护栏命中台账投递	3022	仅回环
计量服务（lp-metering）	用量投影与配额重置	3023	仅回环
维护服务（lp-maintenance-service）	会话清扫、留存清理与审计归档	3024	仅回环
通知适配器	把告警与护栏命中投递到已绑定的聊天渠道	9094	不暴露
密钥服务（secret-service）	独立信任域的密钥权威	3033	不暴露
工作负载令牌续期（两个进程）	分别为平台侧与模型网关数据面续期短生存期工作负载令牌	无独立监听端口	不暴露
关系主库（PostgreSQL 16）	平台库、网关库、密钥库；本形态有专属启动参数	5432	仅回环
连接池（PgBouncer）	拆分微服务的事务级池	6432	不暴露

组件（进程名）	在 Zentrix 形态下的职责	端口	暴露
内存态（Redis 7）	限流计数、熔断状态、广播	6379	仅回环
授权服务（SpiceDB）	关系型授权判定	8443 HTTP / 50051 gRPC	不暴露
授权服务独立库	授权服务专属存储	5432	不暴露
对象存储（RustFS）	租户桶与平台桶	9000 / 9001	仅回环
列式库（ClickHouse）	日志、观测事件与原始用量	8123；指标 9363	仅回环（指标不暴露）

说明

运行时数据面前置在本形态下随包部署、随栈启动，职责是保持与控制面之间的签名快照刷新与本地台账，并提供自身的就绪与指标端点。它的三组路由——访问准入子请求、助手模型出口、按身份的 MCP 服务清单——都要求调用凭据里带智能体身份，本书据实记录这一职责范围。

A.2 一次性任务与工具容器

以下 9 个服务不是常驻进程：前 8 个在栈启动时按顺序执行一次并退出，最后 1 个只在运维执行对应动作时启动。

组件	作用	执行时机
配置代际前置校验（zenova-preflight）	比对二进制内嵌登记表摘要与已发布代际记录的摘要，不一致即失败退出；不接任何网络	三个迁移与初始化任务都以它成功完成为条件
平台库迁移	平台关系库 schema 迁移	前置校验通过、主库健康后
授权服务 schema 迁移	把仓库内的授权模型差分迁移到授权服务	授权服务独立库健康后
网关库初始化	模型网关数据面的库与账号准备	前置校验通过、主库健康后
密钥服务迁移	密钥服务库 schema 迁移	前置校验通过、主库健康后
对象存储初始化	平台桶与租户桶的创建与策略写入	对象存储健康后
护栏内置规则播种	写入平台层内置个人信息规则	平台库迁移完成后
护栏模型工件装载	把镜像内模型文件拷入模型卷；不接任何网络	栈启动时
管理员凭据事务工具	两阶段自校验的 root 凭据写入	仅在运维执行该动作时启动

A.3 网络分区与网段默认值

网络	默认网段	覆盖项	说明
平台网（infra）	10.244.1.0/24	`LOBSTERPOOL_INFRA_SUBNET`	动态分配区间默认 10.244.1.128/25（`LOBSTERPOOL_INFRA_IP_RANGE`）；除两个不接任何网络的一次性任务与一个共享入口网络命名空间的进程外，其余服务均在此网
第二张网络（agents）	10.244.2.0/24	`LOBSTERPOOL_AGENTS_SUBNET`	本形态下承载 9 个平台服务的第二张网卡
指标发现网	无固定网段	—	声明为内部网络、无外网出口

网络	默认网段	覆盖项	说明
容器引擎默认地址池	10.244.0.0/16，子网长度 24	`DOCKER_DEFAULT_ADDRESS_POOL_BASE` / `_SIZE`	安装链路统一对齐
入口在平台网的静态地址	10.244.1.10	`LP_CADDY_INFRA_IP`	模型网关数据面的可信代理白名单是它的 /32，两者必须成对修改
网段冲突候选池	10.244.0.0/16 → 10.245.0.0/16 → 172.28.0.0/16 → 172.29.0.0/16 → 100.96.0.0/16 → 100.97.0.0/16	—	与宿主路由冲突时按序自动避让

A.4 可启用能力与开关速查

下表每一项能力在 Zentrix 形态下都随包就位，通过对应开关按客户的策略启用。**交付时逐项过一遍并记录启用决策**，这张表可以直接当作交付验收清单使用。

能力	启用开关	启用后的效果
每调用方令牌的速率与词元上限	令牌上的速率限制启用标志（逐令牌配置并设定上限）	该令牌的每分钟请求数与词元数被计入真·1 分钟滚动窗口，超限即拒绝
网关策略四族	请求阻断、防重放、词元上限、并发上限各自的启用标志	对应准入判定在数据面生效
熔断与通用限流	由控制面发布含对应动作的统一路由策略	策略下发后在数据面按声明的主体与窗口生效
未注册主机名拒绝	`ZR_GW_BASE_DOMAIN`（填入网关基础域名）	该域名下未注册的主机名在域名层被拒绝
出网 SSRF 守卫（平台侧）	运行时配置项 `ssrf_guard_enabled`，由平台凭证域管理员翻转，10 秒内跨进程生效	云元数据、私网、回环、链路本地、唯一本地与运营商级 NAT 地址被拒绝，并在连接期重解析、把结果固定到具体地址
出网 SSRF 守卫（模型网关数据面侧）	数据面设置项 `ssrf_settings.enabled`	数据面调用上游模型时按同一套拒绝名单判定；未启用时仍要求协议为 http/https 且主机非空
上游在途并发上限	`ZR_UPSTREAM_MAX_IN_FLIGHT`（设为大于 0 的上限值）	超出上限的上游调用先等许可，默认等待 100 毫秒
进程级过载门	显式配置进程级 CPU 与内存阈值	在主机级过载门之外，按 cgroup 归一化的进程水位也参与过载判定
流式重放与断点续传	`ZR_STREAM_REPLAY_ENABLED`，并配置不少于 32 字节的续传密钥	流式响应可在断点后续传
语义续写	`ZR_SEMANTIC_CONTINUATION_ENABLED`（以流式重放已启用为前提）	续传时按语义而非字节边界衔接
gRPC 反射	`ZR_GRPC_REFLECTION`	gRPC 平面在凭据校验之前回答"这里有哪些服务"，用于内网调试
控制面接口规格与交互式文档端点	`LP_OPENAPI_PUBLIC`	规格与文档端点可从平台内部访问；入口对这两个路径仍返回 404
网关模型密钥的所有权转移	`LP_GATEWAY_MODEL_KEY_OWNER_TRANSFER_ENABLED`	租户管理员可转移密钥所有权

能力	启用开关	启用后的效果
授权判定的一致性降级档	`SPICEDB_CONSISTENCY_DEGRADE`	浏览类查询允许低延迟一致性；每次翻转写审计级日志
单点登录身份源	身份源配置的启用标志（按租户配置）	通用 OAuth2 或钉钉登录路径生效
用户即时开通	首次登录自动建号的启用标志	身份源首次登录自动建号，需同时配好落地工作空间与角色
调用方身份头签名	`LP_ACTING_USER_HEADER_JOSE_ENABLED`（控制面与执行面两侧成对开启）	调用方身份头改用双层签名加密封装
内置个人信息识别规则（22 条，四个包）	每条规则各自的启用标志，按客户的合规要求逐条启用	对应类别参与检出与脱敏，动作统一为脱敏
备份保护策略（口径登记，不是执行者）	备份保护策略的启用标志	策略表按填入的口径记录本地与异地的计划与保留（默认每日 02:00、各保留 3 份或 7 天）；**备份集的周期性产出由客户自建的定时任务执行**
异地备份拷贝定时器（随包安装、默认禁用）	`LP_BACKUP_RESTORE_OFFHOST_SCHEDULE_ENABLED`，并具备有效计划、平台密钥与计划令牌	三者齐备才真正调度异地拷贝；缺任一项则定时器保持禁用
恢复演练（随包安装、默认禁用）	演练任务经平台接口创建	演练留在任务历史里并受恢复门禁约束

合规结论

本表就是 Zentrix 形态可启用能力的完整清单，链路导出不在其中：`OTEL_*` 与 `ZR_OTEL_*` 在本包渲染时被逐条移出各服务的配置面。可观测出口只有指标端点、结构化业务事件与容器日志三条，**模型调用正文不经这三条出口离开部署**。

附录 B · 术语表与命名对照

本书对产品概念一律使用中文；代码模块名只在需要精确指代某个进程时出现。同一概念在全书只使用一种写法，下表即该写法的权威定义。

B.1 产品与形态

本书写法	代码 / 英文对应	定义	禁止写法
Zentrix	发行形态 `zentrix-gateway`，`productName: Zentrix`	可独立交付的企业级 AI 网关：模型接入与路由、流量保护、调用治理与凭据分发、内容护栏、调用可观测与计量	Zentrix Router、ZStack Router（对外）、Zentrix 网关平台
Zenova	—	在 Zentrix 之上加装智能体运行时构成的 AI 智能办公私有化平台	Lobsterpool（对外）、Zenova 网关
婉儿（Waner）	waner	Zenova 的智能体运行时模块，加装后 Zentrix 即成为 Zenova（见 1.8）	在 Zentrix 语境下出现（应改写为"Zenova 的可扩展路径"）
交付形态	flavor	Zentrix Gateway、POC、完整（Full）、一体机	版本、套餐、SKU（正文中）
模型网关	—	Zentrix 提供的核心能力名称	大模型网关、LLM 网关（混用）

B.2 进程与平面

本次统一的关键决定，两条，都是合稿阶段留下的撞车：

其一，曾用「数据面网关」指代 lp-gateway、用「网关数据面」指代 zstack-router。两个词只差语序，却指两个职责完全不同的进程，且并排出现在同一张图里。本书取消这两个写法：zstack-router 一律写**模型网关数据面**，lp-gateway 一律写**运行时数据面前置**。

其二，lp-gateway 与 lp-agent-gateway 在各章草稿里各自积累了多个别名（前者出现过智能体接入代理、智能体接入数据面、内部调用代理、平台侧数据面节点、平台策略数据面五种；后者出现过受治理能力执行面、受治理执行面两种）。本书各取一种写法，其余全部列入下表的禁止写法列。

下表即全书唯一权威写法。

本书写法	代码 / 英文对应	定义	禁止写法
模型网关数据面	zstack-router、zr-server	模型调用的数据面进程：准入、路由、协议变换、用量记账。需要精确指代进程时才写进程名	网关数据面、路由器、Router、ZStack Router
控制面	lp-server（容器名 lobsterpool）	平台 API、管理后端，以及受治理 MCP 的对外入口	主服务、后端、Server
能力平面	lp-capability-plane	有效能力投影与绑定治理，直读数据库与授权服务，不回调控制面做鉴权	能力面（与"能力"统称混淆时）

本书写法	代码 / 英文对应	定义	禁止写法
受治理调用执行面	lp-agent-gateway	受治理的 MCP 与连接器单次调用的执行进程，凭据在此跳从已验证租约解密；编排中的服务角色标注为工具网关、等级为关键	智能体网关、Agent Gateway、执行网关、受治理能力执行面、受治理执行面
运行时数据面前置	lp-gateway	与控制面保持快照刷新并持有本地台账的数据面前置。其三组路由均以智能体身份为前提	数据面网关、网关（单用）、LP 网关、智能体接入代理、智能体接入数据面、内部调用代理、平台侧数据面节点、平台策略数据面
入口	Caddy	唯一南北向入口，负责反代与 TLS 终止	反向代理、网关（与模型网关冲突）
护栏	lp-guard-service	内容安全与个人信息保护的统一服务	安全网关、内容审核服务、Guard
密钥服务	secret-service	独立信任域的密钥权威，不吃主密钥	密钥管理中心、KMS
配置代际前置校验	zenova-preflight	比对内嵌登记表摘要与已发布代际摘要的一次性校验进程	健康检查、启动检查

B.3 概念与对象

本书写法	代码 / 英文对应	定义	禁止写法
供应商	provider	模型服务的上游厂商	厂商、提供方、Provider
线路协议	wire protocol	一条上游端点所讲的请求、响应与流式语义	接口协议、API 协议
渠道	channel	网关中一条到上游的接入配置；与即时通讯渠道同名时需消歧	通道
模型服务 / 模型路由	model service / model route	租户侧的模型编排对象	模型配置（混用）
调用方	consumer	持有网关密钥的客户端主体	客户端、用户（在网关语境下）
词元	token	模型计量与限额的单位（用量汇总、每分钟词元上限、词元计数端点）	token（正文中）、令牌（与身份令牌撞车）
调用方令牌	consumer token / API key	调用方持有的网关密钥本体，速率与词元限额挂在它上面	令牌（单用，需带限定语）、API Key
能力	capability	可被治理与调用的工具、技能、连接器、MCP 或插件的统称	功能、技能（作为统称时）
智能体	agent	由平台创建并托管的、有独立容器与工作区的执行体，属于 Zenova 形态；本书只在可扩展路径中提及	Agent、代理、机器人、AI 助手（混用）
签名配置包	gateway bundle	控制面向数据面下发的、由 Ed25519 签名覆盖的配置载荷	配置下发、配置同步
配置代际	runtime generation	一次编译产出的、不可变的运行时配置投影	配置版本、配置快照

本书写法	代码 / 英文对应	定义	禁止写法
租户 / 工作空间 / 部门 / 成员	tenant / workspace / department / member	组织模型的四层结构	组织、团队、Org、Team
默认启用 / 可配置启用	—	全书唯一的能力标注口径	支持、已支持、即将支持、规划中

B.4 写法约定

能力标注：任何能力首次出现时都写明它是默认启用，还是通过开关配置启用。可配置启用的能力必须给出开关名与启用后的影响；给不出开关名的能力不写入本书。可启用能力的完整清单见附录 A.4。

形态口径：本书描述的默认值一律指 Zentrix 交付包的取值。当某项在不同交付形态取值不同时，本书只写 Zentrix 的取值并注明这是形态差异。

数字与单位：端口写裸数字；网段写 CIDR；容量统一使用 GiB 与 MiB 并与代码单位一致；时间统一使用秒、分钟、天，代码中以毫秒表示时在括号内标明；比例写小数而不是百分数，除非代码本身即为百分数。

默认值不一致时两个都写：同一项存在代码默认值与出货配置默认值两个取值时，本书两个都给出，并写明哪一个在 Zentrix 标准交付中实际生效——只写其中一个会误导容量规划与安全评估。

可验证性表达：本书正文不出现文件路径、行号与可直接执行的命令。需要强调某项行为被强制时，采用"由数据库约束强制""在请求期判定""在启动时校验，不符即拒绝启动"这类表述——它们指向可被独立验证的行为，而不是内部实现位置。