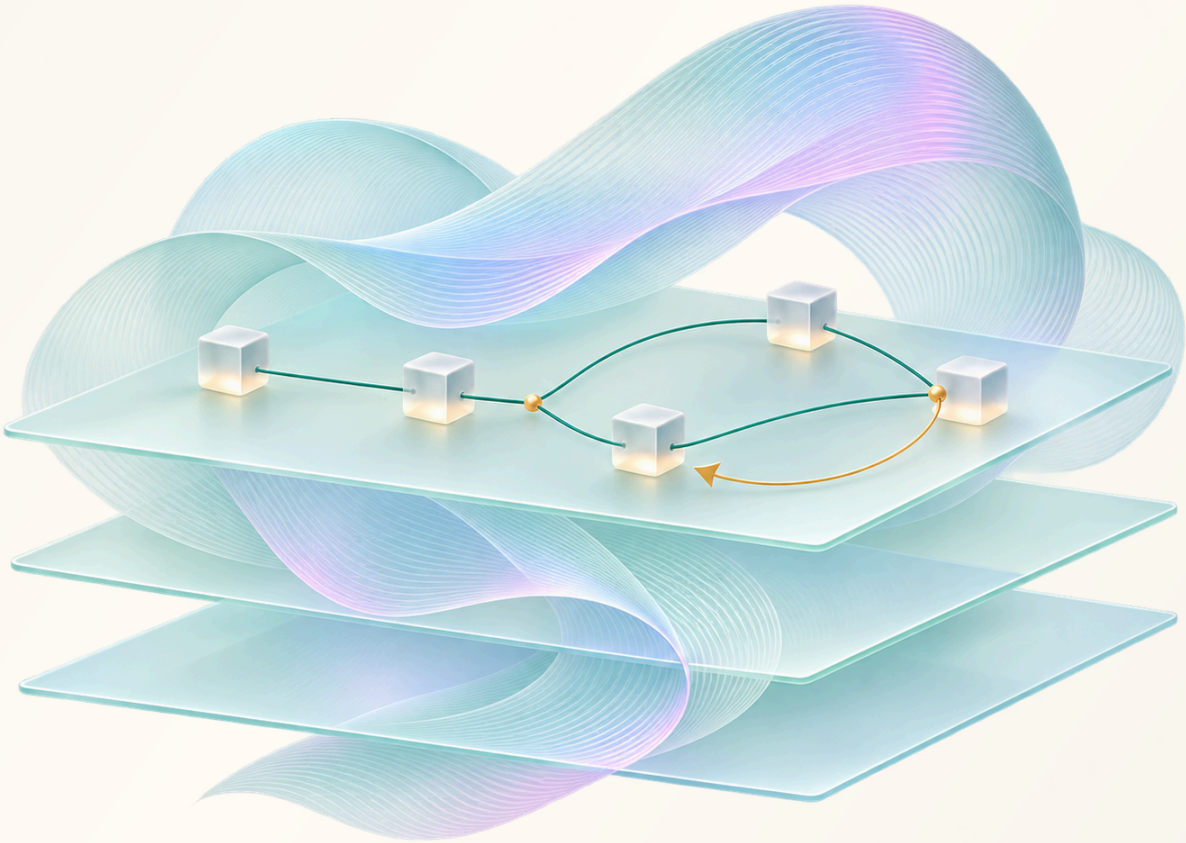
1id1.net

数字出版 / 中文版

云服务与 AI PaaS



CLOUD · AGENT PAAS · ADVANCED AI

使用 Azure 与中国领先云平台构建 AI 原生应用

潘无名

1id1:core:20261122-0001@v1

1id1.net

云服务与 AI PaaS

使用 Azure 与中国领先云平台构建 AI 原生应用

潘无名

简体中文版

版权与版本

云服务与 AI PaaS

作者	潘无名
语言与内容	简体中文 · 六篇 · 二十章 · 126 幅插图
发布平台	1id1.net
出版信息	第一版 · 2026 年 11 月 22 日
本书对象 PID	1id1:core:20261122-0001
本版引用 PID	1id1:core:20261122-0001@v1
对应语言版	英文版 1id1:core:20261122-0002@v1
出版标识	1id1.net 橙红色出版标识 1id1:core:20261122-0003@v1
封面主视觉	悬浮玻璃平台与数学丝带 1id1:core:20261122-0004@v1
AI 使用说明	本书使用 ChatGPT Pro 辅助写作。

© 2026 潘无名
产品名称及商标归各自权利人所有。

前言

模型能力、推理效率与智能体执行机制的演进，使 AI 应用的架构选择深入到基础设施的各个层次。一次任务的质量、时延和成本，受到推理服务、上下文组织、数据访问及执行环境的共同影响；持续执行的任务又把状态、身份与资源变更连接起来。《云服务与 AI PaaS》从这些联系展开，讨论云平台如何参与 AI 系统的构成，以及应用如何在平台能力之上形成自己的运行机制。

平台抽象既影响应用如何构造能力，也决定了工程工作的分配。数据服务、推理接口与执行机制的组合，塑造了应用的交互、工作流与自主执行方式。推理服务的托管程度，影响团队能够介入哪些调度和容量决策；智能体运行时对状态的封装，则影响任务恢复、迁移与版本演进的方式。平台承担什么，应用保留什么，以及两者如何衔接，构成了本书持续讨论的问题。

本书采用资源级思维作为贯穿全篇的分析方法，把架构选择落实到资源的身份、依赖和生命周期，并把控制平面中的配置与变更，同数据平面的执行路径及运行后果联系起来。这一视角既适用于计算、网络和存储，也可以延伸到模型部署、检索索引及智能体平台，同时保留各自不同的管理与执行边界。

因此，书中的基础设施章节直接参与 AI 系统的架构讨论。网络路径与数据位置影响服务之间如何协作，身份与隔离机制限定数据和工具的可达范围，持久状态与交付机制影响任务如何延续。应用能够获得怎样的可靠性、可观测性和成本结构，需要结合这些机制共同判断。云平台对实现细节的封装，使这种判断更多地落在服务之间的关系及其保证范围上。

AI Infra 的进展也使这些关系更值得细究。推理侧的内存管理与请求调度，会影响容量、吞吐与时延之间的取舍；平台对部署方式和计费单位的选择，又会影响这些取舍如何表现为应用成本。本书关注这些实现层面的约束如何通过服务接口、部署形态和计费方式进入应用架构，并据此讨论服务选择、容量规划与系统设计。

沿着这条线索，AI PaaS 与 Agent PaaS 可以放在平台抽象的演进中考察。本书用 Agent PaaS 讨论智能体运行时、会话与任务状态、工具接入，以及评估和运行管理如何被组织为平台能力。这里尤其值得辨明的是，不同状态由谁保存，执行跨越哪些边界，平台能够恢复什么，以及哪些业务后果仍需应用自己处理。会话的延续、任务的恢复与外部操作的补偿，各自需要相应的机制。

当云能力进一步成为智能体工具时，平台也进入了智能体的行动空间。工具接口需要表达操作的对象与范围，执行系统需要落实授权，并处理重试可能带来的重复作用。模型提出的动作、实际发出的调用与资源最终发生的变化，应当能够联系起来考察。由此，控制平面、工具设计和智能体运行时之间形成了一项具体的工程工作：把模型的行动选择接入有明确权限和执行语义的系统。

评估、可观测性和 FinOps 贯穿这些选择。模型、提示、检索索引或工具实现的变更，都可能改变同一任务的执行轨迹。判断一次变更是否值得发布，需要把任务质量与运行记录联系起来，并考察相应的时延、资源消耗和费用。这样，模型效果、系统行为与运行成本可以进入同一次工程决策，平台选择也有了可以持续修正的依据。

我选择 Azure 作为主要叙述参照，使资源模型、身份体系与平台服务能够在同一套实现中连续展开。阿里云、腾讯云和火山引擎提供了必要的比较视角。同一项工程目标，在不同平台的产品边界、服务组合与运行机制下，可能形成不同的架构。比较的意义在于辨明这些差异会怎样影响应用设计，以及哪些判断可以迁移到另一个平台。

全书从资源组织与治理展开，经过应用平台、容器、Kubernetes 和持续交付，进入可观测性、成本管理与 AI 平台，最后讨论检索、智能体平台和云能力的工具化。这一编排为不同层次之间的联系提供了连续的参照。各章的具体实验承担验证机制与边界的作用，读者可以根据自己的系统问题，在相关章节之间往返。

模型和平台会继续演进，系统也需要为这种变化保留空间。我希望本书提供一套能够进入设计讨论的分析方法，使我们在采用新的模型或平台时，也能说明它将怎样改变应用的构成、执行方式与资源需求，以及现有系统如何容纳这些变化。把模型能力、平台提供的保证与应用的实际运行联系起来，是本书组织这些内容的出发点。

总目录

前言 iii

第一篇
作为可编程控制平面的云服务 viii

01 云服务作为可编程控制平面 1
02 资源级思维：控制平面、数据平面与生命周期 20
03 多云架构：责任共担、平台基因与工程意图 38

第二篇
基础云资源控制平面 56

04 资源组织、标签与基础设施即代码 57
05 身份与最小权限 86
06 计算、弹性与可靠性 113
07 网络、私有连接与暴露控制 142
08 存储、数据湖与事件驱动的数据路径 174

第三篇
应用与云原生控制平面 201

09 应用 PaaS、函数、事件与 API 管理 202
10 容器、注册表与软件供应链 235
11 托管 Kubernetes 与应用定义 267
12 持续交付、GitOps 与变更治理 306

总目录（续）

第四篇

运维与价值控制平面 335

13 作为托管反馈回路的可观测性 336

14 作为云控制平面的 FinOps 363

15 安全态势、合规与组织治理 391

第五篇

AI PaaS 控制平面 418

16 从数据与 AI 平台和机器学习 PaaS 到 AI PaaS 419

17 从模型 API 到 AI PaaS 资源 438

18 RAG 不是向量数据库 466

19 作为受治理云资源的智能体平台 493

第六篇

从云能力到受治理智能体工具 525

20 将云能力转化为智能体工具 526

主题索引 551

作为可编程控制平面的云服务

一次部署请求经过哪些检查，最终改变了哪个资源？从控制平面与数据平面的分工出发，追踪一个资源的身份、权限和生命周期，再把同一个工程目标放到不同云平台上，比较责任如何划分。

01 云服务作为可编程控制平面

02 资源级思维：控制平面、数据平面与生命周期

03 多云架构：责任共担、平台基因与工程意图

第一篇 / 第 1 章

云服务作为可编程控制平面

为何工业化软件工程始于托管资源

当基础设施、平台、数据和 AI 能力以可寻址资源的形式提供，并具备身份、策略、生命周期、遥测和成本属性时，云计算才成为一门工程学科。

本章内容

- 为何远程服务器这一比喻不足以描述现代云工程
- 云计算的基本特征如何转化为控制平面义务
- 控制平面操作与数据平面操作有何不同
- 为何可部署的应用是资源图，而不是一台服务器
- 云服务如何在 IaaS、PaaS 和 AI PaaS 中将运维责任产品化
- 为何 Azure 能为教学和比较提供连贯一致的参考控制平面
- 如何在不强行建立错误对应关系的前提下，将工程意图映射到阿里云、腾讯云和火山引擎
- 如何在独立能力检查中查看资源图并追踪生命周期事件

询问一位刚接触云的用户“云是什么”，常得到“别人的计算机”之类的回答。这句话之所以令人印象深刻，是因为它戳破了营销话术；但作为工程模型，它并不完整。它说明了硬件位于何处、由谁拥有，却几乎没有解释一种云能力如何被请求、授权、连接、观测、变更、计费或退役。

按小时租用虚拟机固然有用；但如果一台虚拟机被表示为具有明确类型的资源，通过 API 创建，关联网络策略，被赋予身份，可在资源图中查询，受锁保护，通过遥测度量，归属某个成本中心，并通过有记录的生命周期操作删除，它就有了更深的意义：它已经成为工业化软件工程系统的一部分。

这个系统就是云控制平面。控制平面由管理 API、资源模型、身份检查、策略、生命周期操作、清单服务、活动记录、配额和成本范围组成，云能力正是通过它们变得可治理。门户只是该系统的一个客户端；命令行工具、SDK、基础设施即代码引擎、部署流水线以及未来的智能体，都是其他客户端。

这种区别对 AI 原生应用尤为重要。模型终结点本身只是一项可调用能力；而具备资源身份、配额、网络边界、监控、评估记录、策略和成本范围的模型部署，则属于 AI PaaS 控制平面。知识库、检索系统、工具和智能体也遵循同样的演进路径。

本章确立全书的核心观点：应将云服务理解为面向软件工程的可编程控制平面。后续章节将形式化资源级契约，考察基础和应用控制平面，并把这一模型扩展到数据、模型、知识和智能体平台。

本章论点

云的价值并非始于远程硬件，而是始于运维责任被呈现为可编程、可治理且经济上可见的资源生命周期。

工程意图

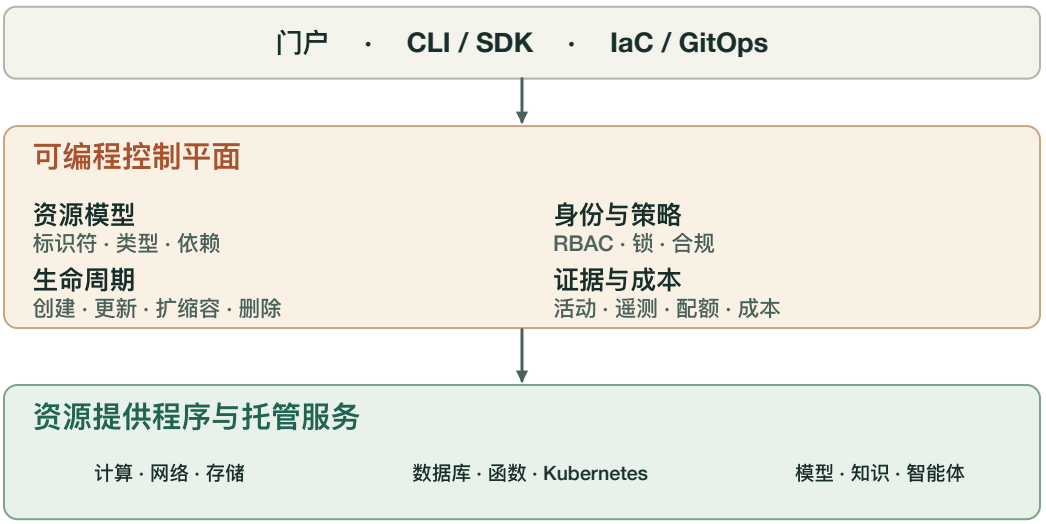


图 1-1 工程意图通过可编程控制平面进入云，而不只是通过产品目录。

1.1 服务器租用的叙事过于狭窄

服务器租用模型是一个有用的历史起点。它说明团队为何无需购买机架、协商数据中心空间或安装虚拟化管理程序，就能获得计算资源；也从基础层面解释了弹性：需要时获取容量，需求下降时释放容量。但如果这一模型让人以为云只是一个更便宜或更快速的数据中心，它就会产生误导。

以一个小型 Web API 为例。若把架构描述为“运行在虚拟机上的应用”，这句话几乎掩盖了所有生产决策：哪个账户拥有这台虚拟机？它位于哪个区域和故障域？选择了哪种镜像和规格？哪些网络接口、子网、路由、安全规则、公网地址和 DNS 记录使其可达？应用凭

借什么身份访问存储或机密？哪条策略禁止公网终结点？哪个监控器检测故障？是否存在备份或重建路径？哪个标签把成本分配给某个团队？哪条操作记录能说明昨晚是谁更改了实例？

因此，虚拟机并不是系统本身，它只是托管资源与责任构成的图中的一个节点。云的工程价值体现在围绕它的四个系统中：

1. 资源系统。能力具有类型、标识符、属性、依赖关系、作用域和生命周期状态。

2. 权限系统。人员、应用、流水线和服务在明确的作用域上获得权限，而不是通过非正式方式访问机器。

3. 面向变更的操作系统。预配、扩缩容、配置、策略评估、审计、删除和恢复均由 API 与控制器协调。

4. 经济系统。使用量能够计量、归属、预算、预测和优化，并与技术所有权关联。

工业级工程并不只由工作负载规模决定。小型系统如果其变更可复现、权限可解释、故障边界已知、证据可检查、成本可归属，同样可以达到工业级；大型系统若这些特性仍依赖没有文档记录的运维人员经验，也可能停留在手工作坊阶段。

控制平面并不等于门户

门户是一种图形化客户端。同一项资源操作应当也能通过 API、CLI、SDK 或声明式部署表达。把屏幕截图视为学习辅助材料，而不是平台的定义。

1.2 从云特征到工程后果

NIST 对云计算的定义至今仍然有用，因为其五项基本特征描述的不只是商业包装，而是一种特定的运行模式：消费者可按需预配；能力可通过网络获得；提供商汇聚资源；容量可以扩展和收缩；使用量能够计量。NIST 还把 IaaS、PaaS 和 SaaS 区分为服务模型，把公有云、私有云、社区云和混合云区分为部署模型。[1] [2]

对工程师而言，每项特征都会带来控制平面层面的后果。按需自助服务需要可编程的请求路径；资源池化需要隔离、放置、配额和调度；快速弹性需要期望状态策略和健康信号；可计量服务需要计量与归属，而不仅仅是月底的一张账单；广泛网络访问则需要明确设计终结点和信任边界。

表 1-1 云特征及其对应的控制平面义务。

云特征	工程后果	成熟控制平面中的证据
按需自助服务	消费者无需提交人工预配工单，即可请求、变更和释放能力。	管理 API、声明式模板、部署记录、生命周期状态

云特征	工程后果	成熟控制平面中的证据
广泛网络访问	必须针对公网、私网、混合以及服务到服务路径设计可达性。	终结点、网络接口、路由、DNS、防火墙或安全组策略
资源池化	共享容量必须维持隔离、放置、配额和所有权边界。	区域/可用区、租户作用域、配额、调度决策、资源标识符
快速弹性	容量成为由策略控制的状态，而不是一次性采购。	扩缩容规则、期望容量、健康信号、活动历史、限制
可计量服务	技术使用量必须对工程负责人和财务负责人都保持透明。	计量项、使用记录、标签、预算、成本分摊、预测、异常

这种转译很重要，因为并非每种托管环境都能提供完整的云运行模式。一台通过多日工单获得、容量固定、没有可编程生命周期且使用量不透明的虚拟化服务器，或许属于远程基础设施，却只提供了部分云体验。NIST SP 500-322 也从运维角度作出了同样的区分：应根据基本特征评估一种计算能力，而不能仅因其部署在本地之外就认定它是云。[2]

最成熟的云服务使这五项特征能够组合。团队可以按需创建存储，通过私有终结点访问，将其置于账户和策略边界之内，使容量独立于虚拟机进行扩展，并把使用量归属到某个产品。服务的价值不仅在于存储字节，还在于它将存储呈现为一份可运维的契约。

1.3 控制平面是工程界面

云操作大体分为两类。控制平面负责管理资源：创建、配置、授权访问、附加策略、检查状态、调整容量或删除资源。数据平面则使用某个资源实例所提供的能力：登录虚拟机、上传 Blob、执行数据库查询、发送事件、调用模型或检索知识。Azure Resource Manager 处理 Azure 的控制平面操作，而数据平面请求通常发往资源专用终结点。[4] [5]

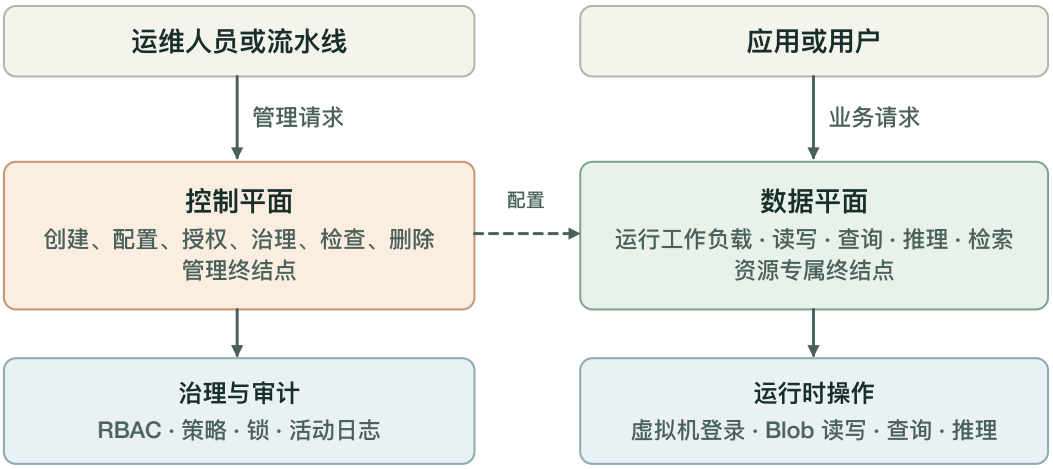


图 1-2 控制平面请求与数据平面请求具有不同的终结点、权限、证据和故障模式。

表 1-2 同一种云服务会公开不同的管理路径和操作路径。

资源	典型控制平面操作	典型数据平面操作
虚拟机	创建虚拟机、选择规格、附加身份、停止或删除资源。	连接来宾操作系统并运行应用。
对象存储	创建账户或存储桶，设置冗余、网络、加密或生命周期策略。	读取、写入、列出或删除对象。
托管数据库	创建实例，选择层级，配置备份、私有访问或故障转移。	执行 SQL 或数据库原生查询与事务。
托管模型	创建模型部署，分配配额、身份、网络、监控或策略。	发送推理请求并接收模型输出。
知识资源	定义数据源、索引架构、摄取、授权、版本和评估。	提交检索请求并接收依据证据。

这种区分可以避免几类常见的安全和可靠性错误。防止删除存储账户的管理锁，本身并不能阻止具有数据平面权限的主体删除 Blob；能够查询数据库的用户未必需要调整或删除数据库资源的权限；模型调用者也不应自动拥有变更模型部署的权限。因此，即使同一身份系统同时参与两者，也必须分别设计控制平面和数据平面的授权。[5]

两个平面的故障方式也不相同。控制平面暂时中断可能会阻止创建新资源，但现有数据平面仍可继续提供流量服务；反过来，管理 API 健康也不能证明应用终结点、数据库或模型推理路径健康。运维证据必须明确正在观测的是哪个平面。

1.3.1 一个管理系统，多种客户端

门户点击、CLI 命令、SDK 调用和基础设施即代码部署，应被理解为管理系统的不同客户端。它们在易用性、可重复性、可审查性和自动化程度上各不相同，但最终都会向资源提供程序请求控制平面操作。

表 1-3 管理客户端及其工程优势。

客户端	最佳用途	主要局限
门户	发现、学习、一次性检查、可视化诊断。	人工表达的意图难以审查、复现、测试和比较差异。
CLI	交互式自动化、调查、脚本编写、可复现命令。	较长的脚本可能变得命令式，并依赖执行顺序。
SDK / REST	产品集成和自定义控制平面应用。	应用必须自行负责重试、状态、授权和 API 版本纪律。
IaC	声明式创建、审查、版本管理、环境复制和受控删除。	并非所有后续日常操作都能自然映射为静态声明。
GitOps / 部署控制器	持续协调已声明的应用或平台状态。	需要明确的事实来源，并谨慎区分生成状态与人工拥有的状态。

ClickOps 本身并非错误，它往往是了解资源模型最快的方式。问题始于门户成为意图的唯一记录。一次生产变更应留下持久证据：经过审查的声明或命令、经过身份验证的管理请求、活动事件、由此形成的资源状态，以及成本或配额后果。

1.3.2 资源提供程序把 API 转化为产品边界

云控制平面并非单体程序，而是把请求路由到资源提供程序。在 Azure 中，Microsoft.Compute、Microsoft.Storage、Microsoft.Network、Microsoft.Web 或 Microsoft.ManagedIdentity 等提供程序命名空间，为某个服务领域定义资源类型和管理操作。Azure Resource Manager 对控制平面请求进行身份验证和授权，然后把请求交给实现该操作的提供程序。[4] [6]

这种提供程序模型解释了云为何既统一又异构。平台可以统一应用身份、策略、标签、活动记录和部署语义，而每项服务仍保留领域专用的架构与生命周期。虚拟机、虚拟网络、搜索服务和模型部署的属性并不相同，但它们参与同一个管理体系。

1.4 应用是资源图，而不是服务器

资源清单很有用，资源图则更为强大。清单回答“存在什么”；图还会捕获包含、附加、依赖、暴露、所有权和证据关系。图把推理单位从孤立产品转变为可运维的系统。

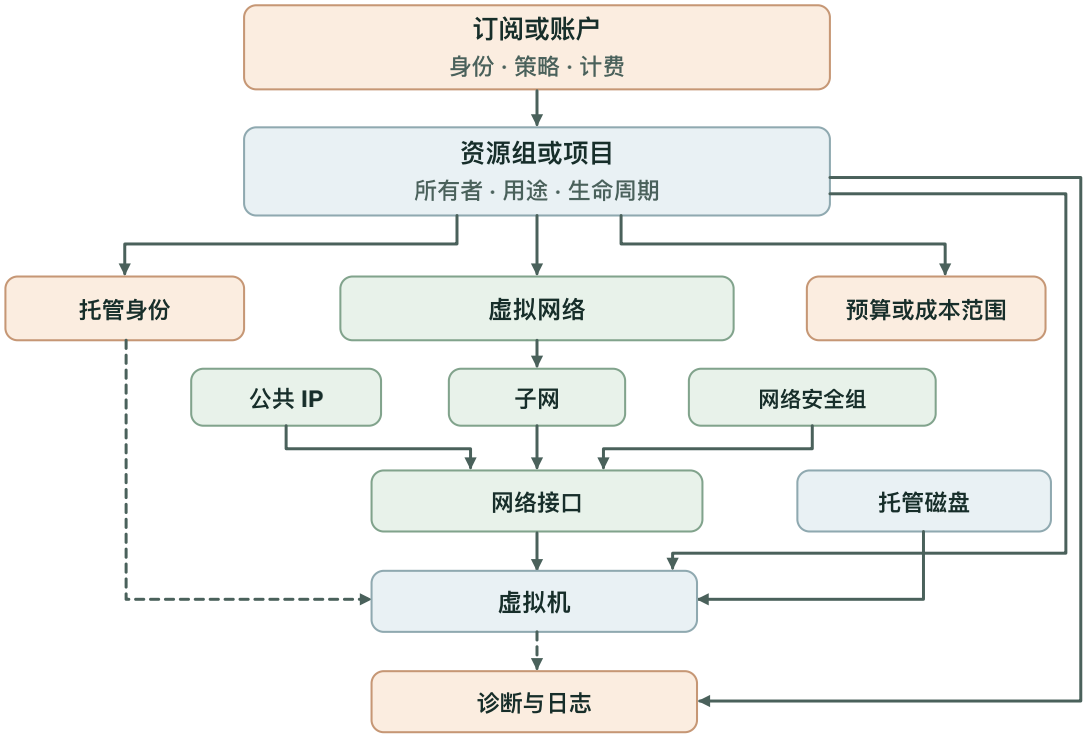


图 1-3 即使是最小的虚拟机应用，也是由计算、网络、身份、可观测性和成本资源构成的图。

资源图能够回答服务器列表无法回答的问题：

- 哪个公网地址、路由、安全规则或负载均衡器暴露了该工作负载？
- 哪个身份能够读取机密、存储对象、模型终结点或数据库？
- 哪些资源共享同一个生命周期，应当一起部署或删除？
- 支出应归属到哪个所有者、环境、应用和成本中心？
- 哪一条活动事件能够解释意外变更？
- 移除主要工作负载后，还有哪些依赖资源残留？

Azure Resource Graph 扩展了 Azure 资源管理，可大规模跨订阅探索资源。它能查询资源类型和属性、联接相关信息，并提供以治理为导向的视图；仅逐个访问服务控制台，很难重建这样的视图。Azure Activity Log 记录订阅级管理事件，并可路由到 Log Analytics、Event Hubs 或 Storage，以便长期分析和保留。[7] [8]

表 1-4 图问题、工程风险与控制平面证据。

问题	未回答时的风险	有用的控制平面证据
谁拥有该资源？	孤立资源、权限不清、成本未分配。	订阅/账户、资源组/项目、标签、创建者、角色分配

问题	未回答时的风险	有用的控制平面证据
谁能访问它?	意外公网暴露或横向移动。	VNet/VPC、子网、终结点、路由、安全组、DNS、流日志
它依赖什么?	删除不完整、隐藏的影响范围、恢复失败。	资源 ID、附加关系、部署堆栈、依赖关系图
谁更改了它?	事件响应缓慢且问责能力薄弱。	活动/审计事件、调用者身份、请求 ID、时间戳、来源
能否重新创建?	配置漂移以及依赖人工的灾难恢复。	IaC 定义、镜像/版本、参数、部署历史
它的成本是多少?	脱离所有权或业务语境的优化。	计量项、标签、计费范围、预算、使用量导出、单位成本计算

PaaS 并未消除资源图，而是压缩了资源图。托管 Web 应用可以隐藏来宾操作系统和修补工作，但仍有托管计划或容量设置、身份、终结点、配置、证书、网络集成、诊断设置、部署历史、配额和成本。无服务器函数隐藏了空闲服务器，却增加了触发器、绑定、并发限制、权限、日志和事件源。托管抽象减少的是客户负责运维的部分，而不是系统必须建模的内容。

1.5 云服务将责任产品化

云中影响最深远的抽象不是虚拟化，而是运维责任的产品化。过去依赖本地流程的任务如今成为服务契约，具备 API、平台自动化、可度量限制和证据。客户仍负责架构决策、应用行为、数据治理，以及服务边界中仍留在客户一侧的配置选择。随着服务模型向上移动，提供商负责更多底层运维工作。

表 1-5 不同服务模型之间的责任转移。

工程关注点	传统方式 / 自主管理	IaaS	应用 PaaS	AI PaaS
物理设施与硬件	客户	提供商	提供商	提供商
虚拟化与基础容量	客户	提供商	提供商	提供商
来宾操作系统与运行时修补	客户	客户	主要由提供商负责	托管运行时由提供商负责
应用代码与配置	客户	客户	客户	客户
数据质量与授权	客户	客户	客户	客户
扩缩容机制	客户自行构建	由客户配置资源	平台能力配合客户策略	平台能力配合模型/智能体策略

工程关注点	传统方式 / 自主管理	IaaS	应用 PaaS	AI PaaS
运维遥测	客户自行构建	云平台 + 客户检测埋点	托管集成 + 应用信号	模型、检索、智能体、工具和应用信号
计量与配额	客户核算	云资源计量项	服务与容量计量项	令牌、吞吐量、检索、评估、工具和运行时计量项

不应把上表理解为高级服务能够消除工程工作的承诺；它们只是改变了工程重心。在 IaaS 中，团队会把更多精力用于操作系统、中间件和实例拓扑；在 PaaS 中，则更多关注服务配置、应用行为、数据契约、身份、策略、部署、可观测性和成本；在 AI PaaS 中，还必须进一步治理模型、提示词、知识、评估、工具权限、安全性以及非确定性行为。

因此，采购托管服务是在购买一份运行契约，而不只是购买一项功能。这份契约包括：提供商自动化哪些工作、客户可以配置什么、适用哪些限制、公开哪些事件与指标、如何升级资源、如何保护数据、如何处理故障、如何计费以及如何退役。优秀的架构会有意识地选择这些契约。

托管不等于无需承担责任

提供商可以自动完成备份、副本、修补或模型托管；客户仍必须选择合适的服务边界、正确配置服务、验证恢复、控制访问、解读遥测，并判断成本和风险是否可以接受。

1.5.1 责任产品化带来的五项工程效应

1. 标准化。资源类型为团队提供共同词汇和可重复的属性，取代只在本地流传的服务器经验。

2. 自动化。生命周期操作成为 API 调用和期望状态声明，可由流水线与控制器执行。

3. 证据。活动记录、健康状态、指标、日志和成本数据，使变更与运行过程可被检查。

4. 规模化运维经济性。提供商可以运营大规模资源群、统一分发修复，并公开若在本地复现会非常昂贵的专业能力。

5. 可逆性。资源可以通过定义明确的生命周期进行版本管理、替换、扩缩容、锁定、恢复或删除，而不再依赖没有文档记录的一连串人工操作。

1.6 以 Azure 作为连贯一致的参考控制平面

本书以 Azure 作为主要实现主线，因为它的管理层次结构和横向服务使控制平面模型格外明确。Azure Resource Manager 提供用于创建、更新、删除、组织资源并控制资源访问的部署与管理层。声明式 ARM 模板和 Bicep 文件描述预期资源，而不是一连串构建机器的

命令。[4]

从高层看，该层次结构依次包括租户与管理组、订阅、资源组、资源提供程序和资源。订阅是授权、配额和计费边界；资源组是面向生命周期、用于容纳相关资源的容器；资源提供程序提供一种或多种资源类型及其管理操作；资源本身则在 Azure 环境中拥有采用全局统一结构的标识符。

```
/subscriptions/{subscriptionId}
/resourceGroups/{resourceGroupName}
/providers/Microsoft.Storage/storageAccounts/{accountName}
```

标识符不只是一条路径，它还是授权作用域、策略评估、活动记录、部署历史、锁、清单查询和成本归属的锚点。设计良好的云系统能够以这个锚点为基础解释每一项重要资源。

表 1-6 使提供商控制平面可见的 Azure 服务。

控制平面关注点	Azure 锚点	工程目的
资源生命周期	Azure Resource Manager 与资源提供程序	创建、更新、检查和删除具有明确类型的资源。
组织	管理组、订阅、资源组、标签	表达所有权、环境、委派和成本范围。
授权	Microsoft Entra 身份与 Azure RBAC	在明确作用域上授予管理操作权限。
防护规则	Azure Policy 与管理锁	审计或拒绝不合规状态，并减少意外变更。
清单	Azure Resource Graph	跨作用域查询资源及其属性。
变更证据	Azure Activity Log	识别调用者、操作、时间、结果和受影响的资源。
经济性	成本管理、预算、计量项、配额	把使用量和支出关联到技术与组织所有权。

这种一致性很有教学价值。无论资源是虚拟机、存储账户、Kubernetes 集群、搜索服务还是 AI 平台对象，都会反复遇到同样的问题：它在何处寻址？谁能管理它？哪项策略适用？它处于哪个网络边界？如何观测？成本是多少？变更或删除时会发生什么？

选择 Azure 并不是因为其他云都应模仿它，也不是因为每项能力都有完全对应的等价物，而是把它作为一套稳定的参考词汇。随后再依据工程意图和证据评估对比平台，而不是进行表面的产品名称匹配。

1.7 在中国领先云平台之间转译控制平面

如果多云比较沦为四列同义词清单，就会失败。名称相近并不保证作用域、生命周期、授权粒度、网络行为、可观测性或计费方式等价。有用的转译应从管理问题出发：资源由什么拥有，如何分组，谁可以对其执行操作，操作记录在何处，如何查询，以及成本如何归属？

目前，阿里云资源管理整合了资源目录、资源组、资源共享、标签和资源中心。资源目录用于组织账户，资源组用于组织单个账户内的资源，标签提供横向分类，资源中心则提供跨账户、跨产品和跨地域的资源视图。ROS 使用模板定义资源及其依赖关系并自动执行部署。ActionTrail 记录管理事件，并可在配置后记录数据事件。[10] [11] [12]

腾讯云使用 CAM 管理身份、角色和权限策略，使用腾讯云企业组织管理多账户结构，使用项目与标签组织资源，并使用 CloudAudit 留存管理活动证据。基于标签的授权和成本分摊，使标签成为治理模型的一部分，而不只是装饰性元数据。[13] [14] [15]

火山引擎文档提供了基于项目的资源管理、IAM 授权、标签、Terraform 与 Pulumi 集成以及 CloudTrail 日志采集。具体覆盖范围因服务而异，而这本身就是一项重要的比较结果：应逐项能力验证资源级管理，而不能仅凭品牌层面的整体承诺作出假设。[16] [17] [18]

表 1-7 在四个参考平台之间转译常见控制平面意图。

工程意图	Azure	阿里云	腾讯云	火山引擎
组织账户	租户 / 管理组 / 订阅	资源目录 / 文件夹 / 成员	腾讯云企业组织 / 部门 / 成员	账户与组织能力；验证当前作用域
资源分组	资源组与标签	资源组与标签	项目与标签	项目与标签
管理授权	Microsoft Entra ID + Azure RBAC	RAM 与资源目录控制策略	CAM 用户、角色、策略、权限边界	IAM 用户、角色和策略
声明基础设施	ARM / Bicep / Terraform	ROS / Terraform	Terraform 提供程序与服务 API	Terraform / Pulumi / 服务 API
查询清单	Azure Resource Graph	资源中心与高级搜索	项目/标签/服务清单；组织视图	项目/标签/服务清单
追踪变更	Azure Activity Log	ActionTrail	CloudAudit	CloudTrail 日志与服务审计记录
归属成本	订阅、资源组、标签、预算	账户、资源组、标签、计费数据	账户、项目/标签、分摊标签、计费中心	账户、项目/标签、计费与配额视图

表中刻意使用了谨慎措辞。比较应当记录不等价之处：某个提供商可能把组织范围清单作为一等服务提供，另一个则通过多个控制台或 API 提供；某项服务可能支持资源级授权，另一项则止于操作或服务作用域；某种审计产品可能默认记录管理事件，而数据事件需要显式配置。这些是工程差异，不是文档上的不便。

第三章将通过责任共担、部署位置、运维边界、集成程度和平台基因进一步阐述这种比较方法。此处的核心原则很简单：先转译工程意图，再检查满足该意图的真实资源与策略模型。

1.8 控制平面的组合

大型云系统几乎从来不只有一个控制平面。控制平面会分层组合：提供商控制平面创建和治理云资源；托管服务控制平面治理服务内部的领域对象；应用控制平面治理业务配置、租户、工作流和功能策略；最终，操作数据平面在这些控制平面建立的状态之下执行请求。

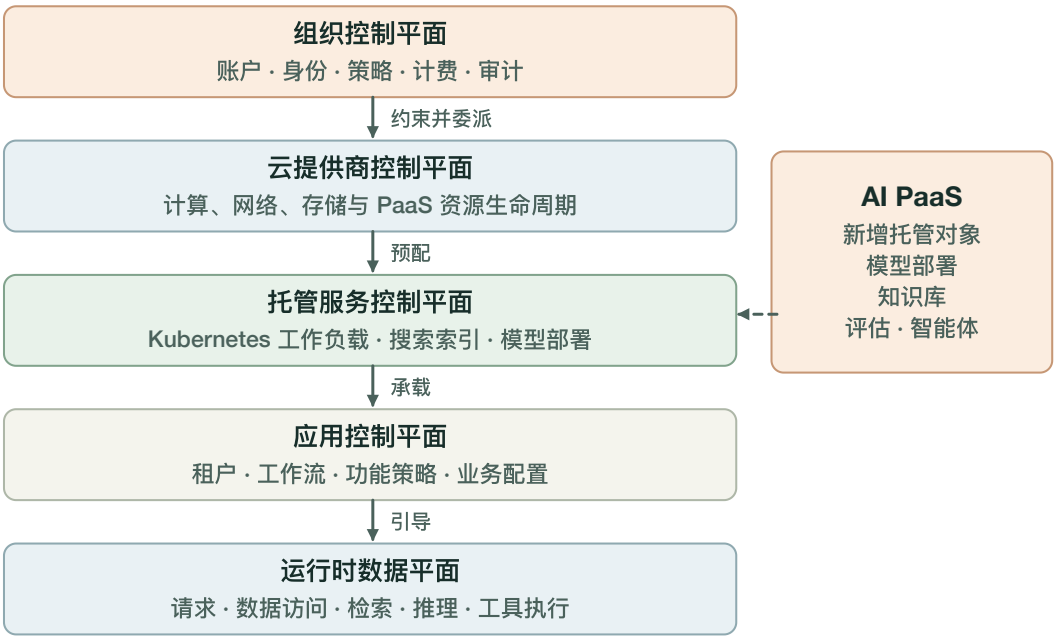


图 1-4 提供商、托管服务和应用控制平面彼此组合，而不是相互替代。

托管 Kubernetes 的例子可以让这些层次变得具体。Azure Resource Manager 创建 AKS 资源，附加身份与网络，选择节点池并记录管理操作。Kubernetes API 是面向 Deployment、Service、Secret 等期望状态对象的服务控制平面。集群上运行的应用还可能公开另一个控制平面，用来管理租户、工作流或业务配置。混淆这些层次会造成权限错配和审计链不完整。

AI PaaS 中也有同样的组合。云控制平面创建 Foundry 或模型资源，并应用 RBAC、网络、策略和成本范围；AI 平台控制平面管理模型部署、知识对象、评估、智能体、工具和运行时配置；应用则控制用户会话、业务权限、审批工作流和产品行为。推理、检索与工具执行发生在数据平面。

1.8.1 AI PaaS 是云工程在控制平面上的延伸

直接模型 API 可能很有价值，但仅有一个可调用终结点还称不上 AI 工程平台。当模型和智能体能力遵循与其他云资源相同的规则——可寻址性、身份、网络边界、生命周期、策略、遥测、配额、评估、成本和退役——AI PaaS 才真正形成。

表 1-8 将云控制平面原则扩展到 AI PaaS。

控制平面原则	基础设施 / PaaS 示例	AI PaaS 示例
资源身份	虚拟机、数据库、函数、Kubernetes 集群	模型部署、知识库、评估、智能体
生命周期	创建、更新、扩缩容、修补、故障转移、删除	部署模型、修订提示词/工具策略、知识版本管理、晋升或退役智能体
授权	谁可以管理或使用资源	谁可以部署模型、检索知识、运行评估或调用工具
网络	公有/私有终结点、对等互连、防火墙	私有推理、私有检索、连接器与工具边界
可观测性	健康状况、延迟、错误、饱和度、日志	令牌、延迟、依据性、工具追踪、评估、安全与质量信号
经济性	实例小时数、存储、请求、传输	令牌、预留吞吐量、检索、向量化、评估、工具与运行时成本
治理	策略、锁、配置合规性	获批模型、安全策略、评估门禁、数据与工具限制

Microsoft Foundry 目前体现了这一方向：它把智能体、模型和工具汇集起来，并配套 RBAC、网络、策略、追踪、监控和评估等企业控制。产品界面会继续演进，但其架构信号是持久的：AI 能力正从库对象和孤立 API 调用进入云资源系统。[9]

本书顺着这一趋势展开。前四部分为资源、应用、交付、运维和成本构建控制平面基础；第五部分把同样的问题应用到数据与 AI 平台、托管模型资源、RAG 和智能体平台；最后一部分说明，如何在不向智能体授予不受限制的平台权限的前提下，把经过验证的云能力公开为受治理的智能体工具。

1.9 能力检查 C1：查看资源图并追踪生命周期事件

这项能力检查有意保持较小范围。它不部署应用，而是通过创建一个低成本资源、从管理平面观察它并收集生命周期证据，来验证本章的核心主张。该练习与后续各实验相互独立。

表 1-9 能力检查 C1 概览。

项目	定义
能力目标	证明云资源具有地址、类型、作用域、属性、标签、生命周期状态、审计事件和清理路径。
Azure 资源	一个资源组和一个不含应用数据的 Standard_LRS 存储账户。
权限	实验资源组的参与者权限或沙盒订阅权限；读取 Activity Log 的权限。
成本边界	不存储数据且在验证后立即删除资源组时，成本可忽略不计。
证据	资源 ID、资源类型、位置、标签、预配状态、创建事件、调用者、时间戳和删除验证。
安全边界	使用沙盒订阅。不要在生产资源组中运行这些命令。

1.9.1 前提条件

- 已安装 Azure CLI，并已使用 az login 完成身份验证。
- 已使用 az account set --subscription <id-or-name> 选择沙盒订阅。
- 具有创建资源组和存储账户以及读取订阅活动事件的权限。
- 已安装 Azure Resource Graph CLI 扩展；如有需要，使用 az extension add --name resource-graph 安装。

1.9.2 创建一个具有明确作用域的资源图

选择订阅中可用的区域。以下命令创建名称唯一的资源组和存储账户，并应用用于表达用途和生命周期的标签。如果其他区域更合适，请替换 eastus。

```
suffix=$(openssl rand -hex 4)
rg=cp-lab-$suffix
location=eastus
storage=cplab${suffix}

az group create \
  --name "$rg" \
  --location "$location" \
  --tags purpose=book-capability-check lifecycle=ephemeral owner=reader
```

```
az storage account create \
  --name "$storage" \
  --resource-group "$rg" \
  --location "$location" \
  --sku Standard_LRS \
  --kind StorageV2 \
  --https-only true \
  --min-tls-version TLS1_2 \
  --tags purpose=book-capability-check lifecycle=ephemeral owner=reader
```

这组命令虽然仍是命令式的，但由此产生的资源并不是匿名副作用。它们具有提供商定义的类型、资源 ID、属性、标签和生命周期状态。后续实验将使用 IaC，使声明可审查、可重复；本次检查则专注于控制平面的可见性。

1.9.3 检查可寻址资源

```
az resource list \
  --resource-group "$rg" \
  --query "[].{name:name,type:type,id:id,location:location,tags:tags}" \
  --output jsonc
```

记录存储账户的资源 ID。确认其中包含订阅、资源组、提供程序命名空间、资源类型和资源名称。然后在订阅作用域查询资源图：

```
az graph query -q "Resources
| where resourceGroup =~ '$rg'
| project id, name, type, location, tags,
provisioningState=tostring(properties.provisioningState)" \
  --output table
```

结果应显示一个存储账户。Resource Graph 可能不会公开每项服务属性，而且摄取可能略有延迟。这些限制也是控制平面契约的一部分：清单查询是管理投影，而不是数据平面本身。

1.9.4 追踪创建事件

```
az monitor activity-log list \
  --resource-group "$rg" \
  --offset 2h \
  --max-events 30 \
  --query "[].
```

```
{time:eventTimestamp,caller:caller,operation:operationName.localizedValue,status
:status.localizedValue,resource:resourceId,correlation:correlationId}" \
  --output table
```

找到存储账户成功的创建或更新操作。记录调用者、操作名称、时间戳、资源 ID、状态和相关性 ID。当一次部署产生多个提供程序操作时，相关性 ID 尤其有用。

活动事件并不是完整证据。它能证明管理操作已经发生，却不能证明应用可以读写对象、网络策略正确，或数据平面授权安全。这些方面需要单独的证据。

1.9.5 验证资源契约

表 1-10 能力检查的证据清单。

问题	预期证据
资源是否可寻址?	完整的 Azure 资源 ID。
能否说出其类型?	Microsoft.Storage/storageAccounts.
能否解释其作用域?	为本次检查创建的订阅和资源组。
能否归属其意图?	purpose、lifecycle 和 owner 标签。
能否检查当前状态?	来自管理平面的资源属性和预配状态。
能否追踪变更?	包含调用者、时间、操作、资源 ID 和相关性 ID 的成功 Activity Log 事件。
能否安全移除?	删除实验资源组，并验证该资源组已不存在。

1.9.6 清理

```
az group delete --name "$rg" --yes

az group exists --name "$rg"
```

最后一条命令应返回 false。删除资源组展示了一条生命周期边界：临时资源被有意组织在一起，从而可以用一次经过审查的操作将它们一并退役。即使资源已经消失，Activity Log 仍会按照平台的保留与导出配置继续保留。

1.9.7 多云转译

表 1-11 依据工程意图重复执行同一项能力检查。

意图	阿里云	腾讯云	火山引擎
创建隔离作用域	创建或选择资源组。	使用沙盒账户/项目和标签。	创建或选择项目与标签。
创建一个低成本资源	创建一个空 OSS 存储桶，或资源组支持的另一项临时资源。	创建一个空 COS 存储桶或其他临时资源。	创建一个空 TOS 存储桶或其他临时资源。
检查清单	使用资源中心或服务/资源组视图。	使用项目/标签筛选器和相应的服务清单。	使用项目/标签筛选器和相应的服务清单。
追踪管理变更	查询 ActionTrail 管理事件。	查询 CloudAudit 事件历史。	查询 CloudTrail 或服务审计记录。
删除并验证	释放资源，并在支持时清空/删除资源组。	删除资源并确认它不再出现。	删除资源并确认它不再出现。

目的不是重现完全相同的控制台步骤，而是收集相同类别的证据：地址、作用域、类型、意图元数据、生命周期状态、审计事件和清理结果。任何缺失的证据类别，都是应记录的重要平台或服务局限。

CloudDesk 复用说明

未来的 CloudDesk FinOps 智能体可以通过两个只读工具使用这项能力：cloud.resource.inventory 返回带有提供商中立的身份和所有权字段的资源；cloud.resource.lifecycle.explain 把审计事件与受影响资源及人类可读的变更说明关联起来。本章不实现这些工具，而是建立它们必须保留的云证据。

```
cloud.resource.inventory(scope, filters)
  -> providerResourceId, resourceType, region, ownership, lifecycleState,
costScope
evidence: inventoryTimestamp, sourceApi, requestCorrelationId
```

1.10 后续章节沿用的原则

本书其余部分将反复应用以下原则：

1. 云是一种运行模式，而不是一个位置。只有当预配、资源池化、弹性、网络和计量通过可运维的服务模型公开时，远程硬件才真正成为云工程。

2. 控制平面是主要的工程界面。门户、CLI、SDK、IaC、策略、清单、审计和成本工具应汇聚到可寻址资源与明确的生命周期操作上。

3. 控制平面权限与数据平面权限不同。创建服务与使用服务需要各自独立的终结点、权限、证据和威胁模型。

4. 应用是资源图。计算资源与网络、身份、存储、可观测性、策略和成本资源相连，这些依赖关系共同决定影响范围和可恢复性。

5. 托管服务将责任产品化。托管服务减少客户负责运维的层次，同时提高配置、数据、身份、策略、证据和服务边界判断的重要性。

6. 多云比较始于意图。先比较所有权、作用域、生命周期、授权、审计、清单和成本，再比较产品名称。

7. 控制平面可以组合。提供商、服务和应用控制平面应具有明确边界，而不是共享一个含义模糊的管理员界面。

8. AI PaaS 延伸了同一套工程原则。当模型、知识、评估、工具和智能体的生命周期、权限、证据与成本受到治理时，它们才成为生产资源。

下一章会把这一宽泛论点转化为精确的资源级检查清单。它将追问：什么使一种能力成为可治理的云资源，而不只是库调用、SDK 对象或不透明的远程服务？随后，本书会用该清单考察每一种基础设施、平台、数据、AI 和智能体能力。

参考资料与官方文档

产品和文档状态核查于 2026 年 8 月 24 日。最终定稿前，应重新核验经常变化的门户路径、API 版本、配额和区域可用性。

[1] NIST SP 800-145：《**NIST 云计算定义**》。五项基本特征、三种服务模型和四种部署模型。https://csrc.nist.gov/pubs/sp/800/145/final

[2] NIST SP 500-322：《**依据 NIST SP 800-145 评估云计算服务**》。用于判断一种能力是否符合 NIST 云模型的运维指南。 https://www.nist.gov/publications/evaluation-cloud-computing-services-based-nist-sp-800-145

[3] NIST SP 500-292：《**NIST 云计算参考架构**》。云系统的参考角色、活动和架构组件。https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf

[4] Microsoft Learn：什么是 **Azure Resource Manager**？Azure 部署与管理层、资源提供程序、声明式语法、标签、访问控制和生命周期管理。 https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview

[5] Microsoft Learn：**控制平面操作与数据平面操作**。管理终结点、操作终结点、授权、锁以及不同平面的特定行为。 https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/control-plane-and-data-plane

[6] Microsoft Learn：**Azure 资源提供程序和类型**。提供程序命名空间、资源类型和管理操作。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/resource-providers-and-types

[7] Microsoft Learn：**Azure Resource Graph** 概述。跨订阅资源探索与面向治理的查询。
https://learn.microsoft.com/en-us/azure/governance/resource-graph/overview

[8] Microsoft Learn：**Azure Monitor** 中的 **Activity Log**。订阅级管理事件及其导出目标。
https://learn.microsoft.com/en-us/azure/azure-monitor/platform/activity-log

[9] Microsoft Learn：什么是 **Microsoft Foundry**？通过企业控制统一管理智能体、模型和工具。
https://learn.microsoft.com/en-us/azure/foundry/what-is-foundry

[10] 阿里云文档：什么是资源管理？资源目录、资源组、资源共享、标签和资源中心。
https://www.alibabacloud.com/help/en/resource-management/product-overview/what-is-resource-management

[11] 阿里云文档：资源编排服务。基于模板的资源定义、依赖关系表达和自动化部署。
https://www.alibabacloud.com/help/en/ros/

[12] 阿里云文档：什么是 **ActionTrail**？管理事件与数据事件、投递、保留、变更追踪及审计用例。
https://www.alibabacloud.com/help/en/actiontrail/product-overview/what-is-actiontrail

[13] 腾讯云文档：**CAM** 概述。针对用户、组、角色、策略和云资源权限的身份与访问管理。
https://www.tencentcloud.com/document/product/598/72360

[14] 腾讯云文档：标签应用场景。资源分类、基于标签的访问控制和成本分摊。
https://www.tencentcloud.com/document/product/651/41690

[15] 腾讯云：**CloudAudit** 产品概述。账户活动、API 使用、资源变更追踪、合规与故障排查。
https://www.tencentcloud.com/product

[16] 火山引擎文档：资源项目管理。基于项目的资源分组，以及对资源组的 IAM 授权。
https://www.volcengine.com/docs/6257/170353?lang=en

[17] 火山引擎文档：**Terraform**。基础设施即代码支持与提供程序配置指南。
https://www.volcengine.com/docs/6706/101989?lang=en

[18] 火山引擎文档：采集 **CloudTrail** 日志。将云操作审计日志采集到 Torch Log Service。
https://www.volcengine.com/docs/6470/2201696?lang=en

第一篇 / 第 2 章

资源级思维：控制平面、数据平面与生命周期

受治理云能力的形式化契约

当一种云能力具备持久身份、管理契约、操作终结点、可观测状态、可问责证据，以及明确的变更和退役语义时，它才变得可治理。

本章内容

- 为何“资源级”是一项架构检验，而不是营销标签
- 持久资源身份与客户端对象、会话、终结点或进程有何不同
- 控制平面契约和数据平面契约如何分离权限、可用性与证据
- 为何必须分别观测期望状态、预配状态和运行状态
- 资源图如何表达依赖关系，同时不强迫所有组件共用一个生命周期边界
- 创建、配置、运行、变更、恢复、分离、保留、删除和清除如何构成托管生命周期
- 为何更新可能意味着就地变更、重启、滚动变更、重新部署、替换或迁移
- 异步操作、重试、相关性标识符和收敛如何影响自动化
- 同一份资源级检查清单如何应用于 IaaS、应用 PaaS、AI PaaS、知识平台和智能体平台
- 在接受一种云能力为可治理能力之前，如何进行简明的资源级审查

第一章把云定义为可编程控制平面，本章将精确阐明这一观点。“资源”一词在技术领域中使用得很宽泛：CPU 是资源，文件句柄是资源，数据库连接是资源，软件库也可能公开一个 Resource 类。这些含义都成立，但对云架构而言过于宽泛。在本书中，“资源级”具有更窄、要求也更高的含义。

资源级云能力拥有持久的管理身份，独立于任何单一客户端进程而存在。它可以通过管理契约被创建或纳管、检查、授权、配置、连接、观测、核算成本、变更和退役；同时还会公开——或治理对其的访问——供应用、运维人员或智能体实际使用的数据平面能力。因此，该资源会成为整体资源环境的一部分，具备所有权、策略、依赖关系、证据和经济后果。

这一定义有意比“可通过 API 使用”更严格。远程 API 可以接受请求，却没有客户可见的生命周期、具有明确作用域的管理身份、策略附加点、清单记录或可靠的退役语义。反过来，高度托管的服务可能只公开很小的数据平面 API，背后却拥有丰富的云控制平面。资源级思维关注的是哪些内容可以治理，而不是开发者能调用多少个方法。

这一区分对 AI 原生系统尤其重要。模型客户端对象不是模型部署；向量集合不会自动成为受治理的知识资源；提示词与工具构成的循环也不会自动成为智能体资源。任何能力要作为工业化软件工程的一部分得到信任，其管理身份、权限、状态、证据、成本和生命周期终止行为都必须明确。

本章论点

云资源并不只是因为远程或可调用而成为资源；定义它的是持久身份和可治理生命周期，这一生命周期把管理意图、操作使用、证据、经济性与退役连为一体。



图 2-1 受治理资源围绕一个持久资源实例综合了七项义务。

2.1 资源级思维是一项检验，而不是标签

本书始终把“资源级”用作诊断检验。它并不意味着所有属性都必须显示在同一个门户页面，也不表示所有服务提供同等的粒度。它要判断的是：某项能力是否足够深入地参与持久的云管理系统，从而支持可重复的工程实践和可问责的运行。

检验从独立性开始。如果删除本地对象或关闭网络连接就会让能力消失，那么该对象很可能只是客户端句柄或会话，而不是云资源本身。如果能力持续存在，拥有提供商定义的类型与标识符，可通过清单重新发现，并通过管理终结点接受生命周期操作，它就是候选资源。当身份、策略、网络、遥测、成本和退役都能附加到这个持久对象时，该能力才变得受治理。

NIST 从按需自助服务、广泛网络访问、资源池化、快速弹性和可计量服务五方面定义云计算。其参考架构在物理资源与服务层之间设置资源抽象与控制层。资源级思维把这些提供商侧的抽象转化为软件团队可管理的消费者侧契约。[1] [2] [3]

界面	哪些内容持续存在?	哪些内容可以治理?	资源级结论
库对象	只存在于客户端进程中	本地配置与代码	不是云资源
远程 API 调用	提供商服务存在，但调用者可能没有具名实例	请求身份验证与提供商通用条款	可调用能力；资源状态尚未证实
托管终结点	部署或终结点持续存在	身份、容量、网络、遥测和版本可能可以配置	通常达到资源级，但仍需验证生命周期
受治理的云资源	具有持久提供商身份的具名实例	生命周期、策略、权限、证据、成本、依赖关系与退役	符合本书的资源级检验

资源级不等于完全托管

IaaS 虚拟机虽然由客户管理来宾操作系统，仍然属于资源级；完全托管的 API 可能由提供商高度管理，却不公开客户专属资源实例。服务责任与资源可寻址性彼此相关，但并不相同。

2.2 资源契约

资源契约是云架构师在把一种能力视为受治理系统的一部分之前，至少应能回答的一组问题。这些问题有意跨越多个领域。产品页面可能会解释功能，而契约追问的是该能力如何融入组织、应用、证据链和生命周期终止流程。

该契约可以概括如下：

```
受治理资源 = 身份 + 控制契约 + 数据契约
            + 状态模型 + 证据 + 经济性 + 退役
```

其中任何一项都不足以单独构成资源契约。只有资源 ID 而没有操作终结点，只能说明某个对象可以管理，却未必可以使用；只有数据终结点而没有管理身份，描述的是一项公用能力，而不是客户可治理的实例；缺少生命周期上下文的遥测无法可靠说明是哪一个版本

或配置产生了结果；没有所有权的成本无法指导行动；缺少保留语义的删除，也不是安全的退役设计。

契约维度	问题	最低证据要求
身份	这个实例究竟是什么，由谁提供，其作用域在哪里？	提供程序命名空间、资源类型、资源 ID、父作用域、区域或放置位置
控制契约	如何创建、变更、授权和退役期望配置？	管理终结点、API 版本、权限、策略结果、操作记录
数据契约	如何使用该能力？	服务终结点、协议、调用者身份、请求/事务证据
状态	请求了什么、正在预配什么、当前实际运行什么？	期望属性、预配状态、健康/就绪/质量信号
证据	能否重建管理变更和运行结果？	清单、活动/审计事件、相关性 ID、日志、指标、追踪
经济性	计量、限制、分摊、预算和优化的对象是什么？	计量项、SKU、配额、标签、成本范围、预算与使用记录
退役	管理结束时会发生什么？	删除/分离行为、保留、软删除、清除、凭据撤销

2.3 身份与可寻址性

资源始于可寻址性。管理系统必须能够区分不同实例，并把操作路由到负责该实例的提供程序。在 Azure 中，资源 ID 对订阅、资源组、提供程序命名空间、资源类型和资源名称进行编码。资源提供程序定义 Microsoft.Compute、Microsoft.Storage 或 Microsoft.MachineLearningServices 等命名空间下实现的类型与操作。[4] [6]

具体语法因云而异，但架构组成部分是稳定的：提供商、类型、名称、父作用域、位置或放置方式，以及 API 契约。仅有显示名称还不够。名称可以重复使用或变更，而提供商生成的 ID 才是角色分配、策略评估、依赖关系、审计事件、计费记录和自动化输出所使用的持久引用。

还应把管理身份与服务终结点区分开来。一个存储资源可以拥有一个管理 ID 和多个数据终结点；模型部署可以只有一个控制平面身份，却公开区域推理终结点或私有推理终结点。终结点可以轮换、禁用或置于网关之后，而拥有它的托管资源不发生变化。只保存终结点 URL 的自动化会丢失生命周期和治理上下文。

```
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}
/providers/{providerNamespace}/{resourceType}/{resourceName}
```

身份要素	用途	常见错误
提供商与类型	路由管理操作并定义架构	假定名称相似的产品具有等价的资源契约
资源 ID	为策略、RBAC、证据和自动化提供持久地址	把门户 URL、DNS 名称或显示名称用作规范身份
父作用域	定义所有权、继承、策略和成本上下文	把资源视为独立于订阅/账户/项目层次结构
API 版本	选择管理架构与操作行为	让生成的示例在不知不觉中选择预览版或过时行为
位置或放置方式	约束数据驻留、容量、延迟和服务可用性	假定管理位置与数据位置相同
标签与元数据	在不改变提供商类型的情况下添加业务分类	用标签代替持久资源身份

2.4 控制平面与数据平面是两份独立契约

第一章介绍了控制平面与数据平面的区别。资源级思维增加了一项重要细化：两个平面是附加到同一托管能力上的独立契约。控制平面回答如何定义、授权、连接、扩缩容、观测和退役资源；数据平面回答资源如何提供服务：执行代码、存储对象、回答查询、发布消息、提供推理、检索知识或运行智能体。

Azure 明确记录了这一区别。控制平面请求通过 Azure Resource Manager 路由，而数据平面请求通常发往服务专用终结点。管理锁适用于 Resource Manager 操作，不会自动应用于服务内部操作。同样，一个主体可能可以管理 Key Vault 资源，却不能读取机密；也可能可以使用已存储数据，却无权调整或删除包含这些数据的资源。[4] [5] [14]

独立契约意味着独立的威胁模型、身份、可用性假设、遥测和 SLO。控制平面可以报告创建成功，但服务终结点仍可能由于 DNS、网络、应用配置、数据就绪状态或授权而不可达；反过来，已预配的服务可以在管理平面暂时受损时继续处理请求。只报告一个平面的设计无法确立另一个平面的健康状况。



图 2-2 控制平面管理资源；数据平面通过另一份契约交付资源能力。

资源	控制平面操作	数据平面操作	为何这一区别很重要
虚拟机	创建、设定规格、附加身份、配置网络、删除	运行来宾进程、读写磁盘、接受应用流量	虚拟机管理成功并不能证明来宾系统已就绪
对象存储	创建账户/存储桶，设置冗余、网络与生命周期策略	放入、获取、列出、复制或删除对象	资源锁不会自动保护每个对象
托管数据库	预配引擎，设置层级、备份、网络和维护策略	执行查询和事务	数据库用户无需同时成为数据库资源管理员
模型部署	选择模型/版本/容量/网络/安全策略	提交推理请求	调用者不应自动获得变更部署配置的权限
知识资源	创建架构、摄取、访问、版本与别名策略	检索有依据的证据	索引健康状况与检索质量需要不同的证据
智能体资源	对指令、工具、知识、策略、评估和发布进行版本管理	启动运行、接收事件、调用工具	运行时权限必须窄于平台管理权限

2.5 期望状态、预配状态和运行状态

“状态”一词存在危险的多义性。声明式模板描述期望状态，即所有者希望平台建立的属性；资源提供程序报告预配状态，即管理操作的进度或最终结果；服务公开运行状态，即能力是否就绪、可达、健康、最新，并满足功能和非功能目标。

这些状态相互关联，但不可互换。期望副本数为五是一条指令，不是观测结果；provisioningState 为 Succeeded 表示管理操作已成功达到最终状态，并不能证明应用通过了健康探测，也不能证明新建索引包含预期文档。Azure Resource Manager 操作可以异步执行，客户端可能需要轮询操作 URL，直到提供程序报告 Succeeded、Failed 或 Canceled。[7] [8] [9]

随着服务托管程度提高，这种区分愈发重要。平台可以成功接受模型部署，但容量分配仍在继续；搜索服务可以接受索引器配置，但摄取随后可能在某个文档上失败；智能体定义可以发布，但某个工具连接仍不可用。控制平面证明管理进度，运行证据则证明服务行为。

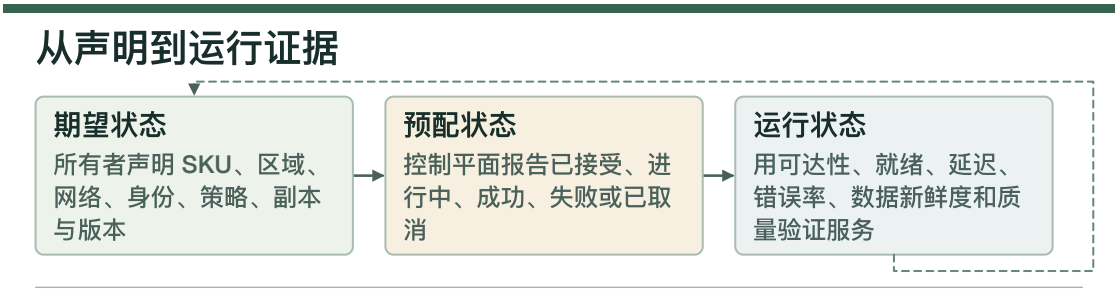


图 2-3 期望状态、预配状态和运行状态构成三条证据通道。

状态通道	负责方	典型问题	证据
期望状态	架构与交付流程	应该存在什么？预期采用哪个版本和配置？	模板、参数、策略、获批计划、Git 提交
预配状态	云资源提供程序与管理客户端	操作是否已接受？是否仍在运行？成功还是失败？	操作 URL、provisioningState、状态码、错误、相关性 ID
运行状态	服务运行时与应用所有者	能力是否可达、就绪、正确、高效，并满足 SLO？	健康/就绪状态、日志、指标、追踪、质量与新鲜度测试

Succeeded 不等于 ready

提供程序可以在底层应用、数据集、路由、证书、模型或索引达到生产可用之前完成管理操作。自动化应等待与依赖项相匹配的证据，而不只是等待通用最终状态。

2.6 资源图与生命周期边界

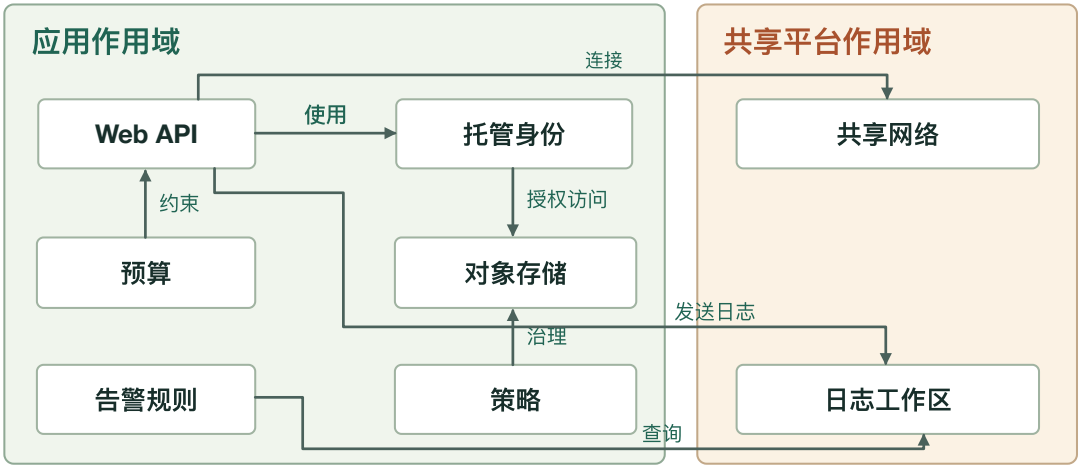
生产解决方案不是单一资源，而是由依赖关系与治理关系连接的类型化资源图。Web 应用可能依赖网络、身份、机密存储、数据库、对象存储、模型部署、监控目标、预算和策略分配。基础设施即代码会显式表达部分依赖关系；其他关系则通过 ID、终结点、角色分配、DNS、标签或运行时配置形成。

Bicep 和 ARM 模板使用引用与 dependsOn 关系确定部署顺序。当一个符号资源引用被另一个资源使用时，Bicep 可以推断出许多依赖关系。这些部署依赖很有用，但完整的运行图更大。预算不必先于 Web 应用创建，也会影响其财务治理；日志工作区可能比发送遥

测的应用存在更久；共享网络可能服务于几十个应用。[10] [11]

资源组是有用的生命周期和管理作用域，但资源位于同一处并不能证明它们都应一起创建、变更和删除。Azure 建议同一资源组中的资源共享生命周期；即使其他提供商的分组原语名称不同，这项原则也同样适用。共享资源应拥有自己的所有权与退役路径，而不是隐藏在某个应用部署中。[4]

应用资源与共享平台有不同的生命周期



资源可以跨域依赖。应用退役时，应显式核对共享网络、日志和保留策略的归属。

图 2-4 解决方案是资源图，其中应用组件与共享平台组件具有不同的生命周期边界。

关系	含义	生命周期影响
拥有	一个资源是子资源的持久父资源	删除或移动父资源可能限制或移除子资源
依赖	创建或运行需要另一个资源	必须明确顺序、就绪状态、回滚和故障传播
附加到	资源已连接但不被拥有，例如磁盘或私有终结点	可以分离而不删除
授权	身份或策略授予对另一个资源的权限	退役时必须撤销角色分配与凭据
路由到	DNS、网关、负载均衡器或事件规则引导流量	切换与回滚可独立于资源替换进行
观测	遥测目标、警报或评估监控另一个资源	证据往往必须比被观测的工作负载保留更久

关系	含义	生命周期影响
将成本分摊给	标签、账户/项目作用域或计费规则分配支出	资源删除后，成本记录仍必须含义明确

2.7 托管生命周期不止于 CRUD

创建、读取、更新和删除是有用的 API 动词，但并不足以构成生命周期模型。工业化云工程还需要验证、策略评估、依赖关系解析、就绪检查、运行、扩缩容、恢复、版本转换、保留、证据保存和退役。把生命周期等同于 CRUD，会让自动化过早宣布成功，也会让删除过于仓促地结束。

生命周期始于预配之前。请求必须通过身份验证和授权；系统可能评估策略、配额、位置、命名、依赖关系和可用容量。提供程序可以接受操作，并异步执行耗时工作。预配之后，资源还必须连接身份、网络、遥测、数据和成本范围；随后，就绪证据必须证明该能力可以安全使用。

运行不是一个被动阶段。资源会消耗配额和预算、发出信号、接收策略变更、经历事件，并可能偏离预期配置。变更可以就地完成，也可以通过替换完成；恢复可以还原受保护状态，也可以创建替代资源；退役必须决定哪些对象、数据、凭据、终结点、证据和财务记录应删除、分离、保留、保持可恢复或彻底清除。

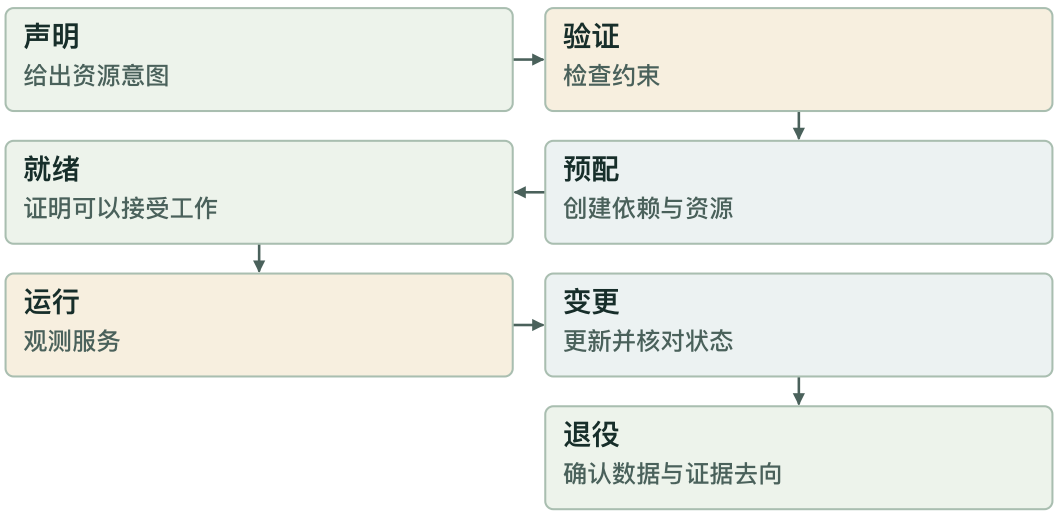


图 2-5 托管资源生命周期包括验证、就绪、运行、恢复和多种退役结果。

生命周期阶段	控制平面责任	验收证据
声明	记录预期类型、属性、依赖关系、所有权和策略	模板、参数、变更记录、获批计划
验证	进行身份验证与授权，并评估策略、配额、名称、区域和架构	验证结果、拒绝原因、配额/容量检查
预配	分配提供商容量并建立资源实例	操作状态、资源 ID、预配状态、错误详情
连接与配置	附加身份、网络、机密、数据源、遥测和预算	依赖关系图、角色分配、终结点与诊断设置
就绪	证明数据路径和功能依赖项可以工作	健康/就绪测试、示例事务、检索或推理测试
运行	观测使用情况、可靠性、安全性、质量和成本	日志、指标、追踪、评估、计费与审计事件
变更与恢复	执行受控变更、发布、替换、回滚、还原或故障转移	计划/差异、版本、相关性、变更前后证据
退役	移除权限，并决定删除、分离、保留、恢复或清除结果	删除/保留记录、已撤销的访问、残留清单与成本检查

2.8 变更语义：“更新”并非只有一种含义

“更新”一词掩盖了多种实质不同的操作。有些属性可以就地修改；有些需要重启进程或滚动重启实例；有些会创建新修订版本，并保留旧修订版本以便回滚；有些会让提供商在同一逻辑名称背后替换资源；还有一些不可变更，必须显式迁移到新的资源身份。

因此，自动化必须理解变更语义，而不只是发送 PATCH 或 PUT。安全的变更计划应明确影响范围、停机时间、数据移动、终结点连续性、身份连续性、成本重叠、回滚路径，以及宣布成功所需的证据。预览和 what-if 能力可以提供帮助，但它们只能报告提供程序与部署引擎所理解的语义。Azure 部署堆栈和 what-if 结果会区分受管理资源的创建、修改、无变更、删除和分离结果。[12] [13]

变更语义也说明了稳定间接层的价值。DNS 名称、别名、部署槽、修订版本、网关路由、索引别名、模型部署名称和工具版本，让客户端可以依赖稳定契约，同时在其背后更改实现。间接层本身也必须作为资源管理，并纳入证据链。

变更形式	变更的对象	主要风险	有用的控制措施
就地变更	现有实例上的属性	意外副作用或不受支持的组合	验证、ETag/并发检查、配置差异

变更形式	变更的对象	主要风险	有用的控制措施
重启	运行时进程或节点以相同资源身份重启	暂时不可用以及内存状态丢失	就绪探测、维护窗口、重试与排空
滚动变更	实例或副本逐步变更	版本混合与容量不完整	最大激增数/不可用数、健康门禁、渐进式验证
修订版本或槽位	在旧版本旁边创建新的不可变版本	流量分配与状态兼容性	别名/路由、金丝雀发布、回滚与版本化遥测
替换	新资源实例取代旧实例	终结点、身份、数据和成本出现不连续	迁移计划、并行运行、切换与退役清单
跨资源迁移	数据或工作负载移动到不同类型、区域、账户或提供商	语义不兼容与证据不完整	导出/导入契约、对账，以及必要时的双读/双写

2.9 删除是一项架构操作

删除经常被当作日常清理。在资源级，它是一项一等架构操作，因为它会改变权限、可达性、保留数据、依赖关系、证据和成本。控制平面删除请求可能只移除管理对象，也可能级联删除子资源，可能被锁或依赖项阻止，还可能把数据置于可恢复状态。数据平面也可能支持独立于容器资源删除操作的应用数据删除。

Azure 管理锁体现了平面边界。CannotDelete 或 ReadOnly 锁会约束 Resource Manager 操作，但不会自动阻止数据平面变更。部署堆栈则引入明确的 actionOnUnmanage 决策：从受管理声明中移除的资源可以删除或分离。分离会保留资源，但终止堆栈对它的所有权。两者结果不同，不能隐藏在同一个“销毁”按钮背后。[12] [14]

退役不应只移除主要资源。角色分配、服务凭据、私有终结点、DNS 记录、事件订阅、诊断设置、仪表板、预算、自动化计划、注册表制品和外部引用都可能残留。某些证据与计费记录还应有意保留。因此，退役验证必须查询残留清单、权限、流量、遥测、数据保留和持续发生的成本。

结果	资源结果	数据/证据结果	适用情形
删除	移除管理对象	可以移除、保留或单独保护	不再需要该能力及其依赖项
分离 / 取消管理	资源保留，但离开某个部署或所有权边界	数据随资源保留	所有权或生命周期移交给另一堆栈/团队
软删除	资源或数据被隐藏，但在一定期限内可恢复	保留到恢复窗口到期	防止意外或恶意删除
保留	有意保留选定的资源、数据或证据	保留遵循法律、运维或审计策略	状态必须比产生它的工作负载保留更久

结果	资源结果	数据/证据结果	适用情形
清除	永久移除可恢复状态	此后无法恢复	保留期和审批条件均已满足
替换	新资源接管责任后退役旧资源	迁移数据/证据或进行对账	不可变更的变更、区域迁移、重新设计或提供商转换

门户干净不等于退役干净

资源从某个控制台消失，并不能证明子资源、访问授权、数据副本、DNS 记录、事件订阅、日志、承诺或收费已经终止。退役需要关于预期残留状态的正面证据。

2.10 异步、重试、幂等性与收敛

控制平面自动化运行在分布式系统之上。请求会经过管理网关、策略引擎、资源提供程序、区域容量管理器和服务后端。成功的 HTTP 响应可能只意味着请求已被接受，而非已经完成。客户端必须区分请求送达、操作接受、资源创建、依赖项就绪和运行成功。

Azure 长时间运行操作会返回供客户端轮询的标头或操作 URL，以及预配状态和最终结果。提供程序实现不尽相同，因此 SDK 轮询器通常比手写的休眠循环更安全。应保留操作记录和相关性标识符作为证据。[7] [8] [15] [16]

重试会带来另一种风险。提供程序接受创建请求后，客户端可能超时；盲目重复非幂等操作可能创建重复对象或相互冲突的工作。提供商可能支持确定性名称、ETag、幂等 PUT 语义或客户端令牌。阿里云 ROS 明确建议在重试时使用 ClientToken，以避免重复操作。[20]

收敛也不是瞬时完成的。清单系统、策略视图、账单导出、DNS、遥测流水线和跨区域副本都可能滞后于权威服务。稳健的工作流会定义：每项决策以哪个来源为准、等待多久、陈旧证据有何特征，以及哪些操作可以安全重试。“最终可见”是一项工程条件，不是忽略验证的借口。

自动化风险	故障模式	必要的设计响应
已接受但未完成	工作流把 HTTP 接受误当作资源就绪	轮询操作，随后执行服务专用的就绪测试
响应丢失	提供程序已接受操作后，客户端再次重试	使用确定性身份、幂等令牌，或在重试前读取并对账
部分失败	资源图的一些节点已存在，另一些节点失败	记录已创建的 ID，检查依赖状态，并选择回滚或继续语义

自动化风险	故障模式	必要的设计响应
并发变更	两个执行者相互覆盖对方的预期属性	使用 ETag/版本检查、适用时的锁，以及串行化所有权
陈旧清单	状态已经改变后，资源仍显示为不存在、存在或不合规	定义权威来源、有界等待和对账循环
已就绪但不健康	管理报告成功，而服务行为失败	把预配证据与运行 SLI、合成测试分开
删除受阻	锁、依赖关系、保留设置或提供程序行为阻止清理	明确显示阻塞项；绝不能在无提示的情况下宣布退役完成

2.11 同一契约适用于各种服务模型

资源级思维有意独立于 IaaS、PaaS 或 AI 术语。随着服务模型上移，提供商可能管理更多底层堆栈，但消费者仍需要可寻址的契约，用于管理仍可配置且需问责的部分。资源身份随之上移：从机器与磁盘，到应用与数据库，再到模型部署、知识库、评估和智能体。

这也是 AI PaaS 不应被描述为“API 加门户”的原因之一。当模型、知识、安全、评估、追踪和智能体配置成为具有独立权限、生命周期、证据和成本的类型化资源时，工程收益才真正出现。资源边界为平台团队提供了可以预配、审查、发布、审计、优化和退役的对象。

服务层	代表性资源	数据平面能力	资源级管理问题
IaaS	虚拟机、磁盘、网络、负载均衡器	运行进程、存储块、路由数据包	规格、放置位置、身份、网络、可用性、替换、备份、成本
应用 PaaS	Web 应用、函数、数据库、消息主题	处理请求、执行函数、查询数据、投递消息	运行时/版本、扩缩容、连接、触发器、部署、健康、保留
云原生平台	集群、命名空间、工作负载、入口、持久卷	调度容器并公开服务	控制器所有权、期望状态、策略、工作负载身份、发布、漂移
AI PaaS	模型部署、知识库、评估、安全策略	推理、检索、评估、筛选	模型/版本、容量、数据边界、质量、延迟、令牌成本、退役
智能体平台	智能体定义/版本、工具连接、运行时、追踪	规划、检索、调用工具、产生结果	权限、记忆、工具风险、评估门禁、审计、成本、回滚、退役

2.12 跨云转译契约

即使云产品名称和资源粒度不同，这份契约仍保持稳定。Azure 通过 Azure Resource Manager、提供程序命名空间、资源 ID、Resource Graph、Activity Log、Bicep 和部署堆栈，使该模型格外可见。阿里云把账户与资源组织同资源管理、ROS 中的资源类型和堆栈、ActionTrail 以及服务专用 API 结合起来。腾讯云使用项目与标签组织、CAM 授权、服务 API 和 CloudAudit。火山引擎公开项目分组、IAM、基础设施即代码提供程序、服务资源 API，以及基于 CloudTrail/TLS 的审计采集。[4] [15] [16] [17] [18] [19] [21] [22] [23] [24] [25] [26] [27]

跨云比较不应强求错误的等价关系。一个提供商可能把某项功能公开为一等资源，另一个则把相同行为置于更大的应用对象内部；一个可能提供可分离的子资源，另一个却要求替换父资源；一个可能通过托管指标公开运行状态，另一个则需要应用探测。比较应记录差异，而不是用通用产品类别掩盖差异。

工程意图	Azure 锚点	阿里云锚点	腾讯云锚点	火山引擎锚点
为资源命名并指定类型	ARM 资源 ID + 提供程序/类型	资源 ID/ARN + 产品/ROS 类型	资源标识符 + 产品 API 类型	资源标识符 + 服务 API 类型
声明资源图	Bicep/ARM/Terraform；部署堆栈	ROS/Terraform	Terraform 与产品 API/模板	Terraform/Pulumi 与产品 API
追踪管理状态	预配状态与长时间运行操作状态	堆栈/资源状态与服务操作状态	服务 API 任务/状态字段	服务 API 任务/状态字段
检查资源环境	Resource Graph	资源中心 / 资源管理	项目、标签、服务清单	项目、标签、服务清单
审计生命周期事件	Activity Log	ActionTrail	CloudAudit	投递到 TLS 的 CloudTrail 日志
终止管理	删除、锁、堆栈删除/分离、服务保留	堆栈删除与服务保留规则	产品专用删除/保留行为	产品专用删除/保留行为

2.13 资源级审查

后续每一章和每个实验都会使用以下审查。它有意保持简洁。团队应能在选择云服务、将其公开为智能体工具或宣布实现达到生产就绪之前完成这项审查。

否定答案不一定意味着服务不合格，而是指出客户需要在哪些位置添加包装器、外部证据、流程控制或更强的隔离边界。重要的是把责任明确下来。

审查问题	预期答案或证据	在以下情形拒绝或升级处理……
1. 持久身份是什么？	提供商、类型、资源 ID、父作用域、位置	只能用客户端对象、URL 或机密标识能力

审查问题	预期答案或证据	在以下情形拒绝或升级处理……
2. 管理终结点是什么?	有文档的控制平面 API/SDK/IaC 契约与版本	生命周期依赖提供商人工干预
3. 数据终结点是什么?	协议、调用者身份、网络路径、请求证据	管理与使用共用未作区分的超级凭据
4. 可以声明哪些期望状态?	类型化属性、默认值、不可变字段、依赖关系	关键配置只存在于无文档记录的门户状态中
5. 如何观测预配?	长时间运行操作、状态、最终错误与相关性 ID	自动化必须依据已经过的时间猜测是否完成
6. 如何证明就绪?	健康、示例事务、质量/新鲜度或合成测试	把预配成功当作生产就绪
7. 什么约束权限与配置?	RBAC/IAM、策略、网络、配额、锁、审批	唯一安全的角色是账户级管理员
8. 依赖关系图是什么?	被拥有、附加、路由、授权、观测以及共享的依赖项	删除或替换的影响未知
9. 变更意味着什么?	就地、重启、发布、修订、替换或迁移语义	更新可能在无提示的情况下替换身份或丢失状态
10. 哪些证据会保留?	清单、活动/审计事件、遥测、版本与成本记录	无法重建事件或变更过程
11. 计量和限制的对象是什么?	SKU、计量项、配额、容量、预算、分摊作用域	成本或容量没有可问责的所有者
12. 如何退役?	删除/分离/保留/恢复/清除，以及凭据和终结点清理	控制台干净是唯一的清理证明

资源级审查练习

选择本书后续会出现的一种能力，例如虚拟机、对象存储、托管 Web 应用、模型部署、搜索索引、知识库或智能体定义。只使用官方文档和账户中可见的证据，完成十二问审查。

不要用单一分数评价该能力。编写一页决策记录，包含三部分：已验证的资源属性；提供商或服务专用的缺口；弥合这些缺口所需的客户控制。保存资源 ID、管理 API 版本、一条成功的管理事件、一项就绪测试，以及有文档说明的退役行为。这是一项设计审查，不是完整实验，不应创建长期存在或昂贵的环境。

决策规则

当一种能力的资源身份和生命周期已经验证、缺口具有明确负责人，并且其操作使用可以与管理和成本证据关联时，它就可以进入受治理架构。

CloudDesk 复用：资源描述与生命周期评估

配套实现工作中描述的 CloudDesk FinOps 智能体，可以复用本章作为能力契约。该智能体绝不能只根据产品名称推断生命周期或权限；它应检索提供商资源身份、检查当前管理状态和运行状态、遍历相关依赖项，并在建议或执行变更前提供证据。

```
capability: cloud.resource.lifecycle.assess
inputs:
  provider: azure | alibaba | tencent | volcengine
  resource_id: string
  requested_operation: inspect | change | replace | retire
outputs:
  resource_type: string
  parent_scope: string
  control_plane_state: object
  operational_evidence: object
  dependencies: array
  change_semantics: in_place | restart | rollout | replace | migrate | unknown
  retirement_semantics: delete | detach | retain | recover | purge | unknown
  evidence_refs: array
risk_controls:
  - read_only_by_default
  - require_plan_for_mutation
  - require_approval_when_identity_or_data_boundary_changes
  - verify_residual_inventory_and_cost_after_retirement
```

这份契约并不实现提供商适配器，而是定义每个适配器都必须返回的内容。这种分离使智能体能够比较不同云，同时不假装它们的 API 或生命周期细节完全相同。

本章小结

- 资源级是一项治理检验：能力必须具有持久身份，以及独立于单一客户端进程的管理生命周期。
- 受治理资源把身份、控制平面契约与数据平面契约、状态、证据、经济性、依赖关系和退役结合在一起。
- 期望状态、预配状态和运行状态回答不同的问题，也需要不同的证据。
- 解决方案是资源图；共享依赖项往往需要与使用它的应用相分离的生命周期边界。
- 生命周期不止于 CRUD，还包括验证、就绪、运行、受控变更、恢复、证据保存与退役。
- 更新可能意味着变更、重启、发布、修订、替换或迁移；自动化必须知道适用哪种语义。
- 删除、分离、软删除、保留、恢复和清除是不同的架构结果。
- 异步操作、重试、陈旧清单和收敛，需要相关性、幂等性、对账和明确的就绪测试。

- 同一份资源契约适用于 IaaS、应用 PaaS、云原生平台、AI PaaS、知识资源和智能体平台。
- 跨云比较应保留资源粒度与生命周期上的差异，而不是强求产品名称等价。

参考资料与官方文档

产品和文档状态核查于 2026 年 8 月 25 日。最终定稿前，应重新核验提供商架构、API 版本、资源类型、生命周期状态、保留行为和区域可用性。

[1] NIST SP 800-145：《NIST 云计算定义》。按需自助服务、广泛网络访问、资源池化、快速弹性和可计量服务。https://csrc.nist.gov/pubs/sp/800/145/final

[2] NIST SP 500-292：《NIST 云计算参考架构》。资源抽象与控制层、服务层、云服务管理、参与者和编排。https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf

[3] NIST SP 500-322：《依据 NIST SP 800-145 评估云计算服务》。检验一种能力是否符合云模型的方法。https://www.nist.gov/publications/evaluation-cloud-computing-services-based-nist-sp-800-145

[4] Microsoft Learn：什么是 Azure Resource Manager？部署与管理层、资源组、声明式管理、身份、标签和生命周期指南。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview

[5] Microsoft Learn：控制平面操作与数据平面操作。各平面专用的终结点、授权、锁和操作行为。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/control-plane-and-data-plane

[6] Microsoft Learn：Azure 资源提供程序和类型。提供程序命名空间、资源类型、注册与操作。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/resource-providers-and-types

[7] Microsoft Learn：使用 Azure SDK for Go 执行控制平面操作。把控制平面定义为控制资源创建、更新、配置和删除的 API。https://learn.microsoft.com/en-us/azure/developer/go/control-plane

[8] Microsoft Learn：异步操作的状态。Azure-AsyncOperation、Location、Retry-After、provisioningState 与最终结果。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/async-operations

[9] Microsoft Learn：Azure REST API 参考。Azure 资源的 REST 终结点、HTTP 操作、身份验证、请求与响应。https://learn.microsoft.com/en-us/rest/api/azure/

[10] Microsoft Learn：在 Bicep 中设置资源依赖关系。隐式与显式依赖关系，以及依赖关系可视化。https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/resource-dependencies

[11] Microsoft Learn：定义 ARM 模板中的资源部署顺序。dependsOn 与部署排序。https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/resource-dependency

[12] Microsoft Learn：使用 Bicep 创建和部署 Azure 部署堆栈。统一资源管理，以及删除/分离 actionOnUnmanage 语义。https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deployment-stacks

[13] Microsoft Learn：使用 what-if 预览部署堆栈变更。创建、修改、无变更、删除与分离结果。https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deployment-stacks-what-if

[14] Microsoft Learn：锁定 Azure 资源以保护基础设施。CannotDelete 和 ReadOnly 管理锁及其作用域行为。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/lock-resources

[15] Microsoft Learn：Azure Resource Graph 概述。跨订阅清单与治理查询。
https://learn.microsoft.com/en-us/azure/governance/resource-graph/overview

[16] Microsoft Learn：Azure Activity Log。订阅级控制平面事件、关联、路由、保留与分析。
https://learn.microsoft.com/en-us/azure/azure-monitor/platform/activity-log

[17] 阿里云文档：什么是资源管理？资源目录、资源组、资源共享、标签和资源中心。
https://www.alibabacloud.com/help/en/resource-management/product-overview/what-is-resource-management

[18] 阿里云文档：ROS 资源类型索引。阿里云产品中的标准资源类型和数据源类型。
https://www.alibabacloud.com/help/en/ros/developer-reference/list-of-resource-types-by-service

[19] 阿里云文档：使用 Ansible ali_ros_stack 部署 LNMP 环境。堆栈创建，以及删除模板所创建的资源。
https://www.alibabacloud.com/help/en/ros/use-cases/use-ansible-ali-ros-stack-to-deploy-a-lnmp-environment

[20] 阿里云文档：ROS SDK 常见问题。使用 ClientToken 避免重试期间产生重复操作的指南。
https://www.alibabacloud.com/help/en/ros/support/what-do-i-do-if-the-errcode-sdk-serverunreachable-error-is-reported-when-i-use-sdks

[21] 腾讯云文档：项目管理。把云产品和服务归入项目作用域进行管理。
https://www.tencentcloud.com/document/product/378/34726

[22] 腾讯云文档：资源标签。资源分类、基于标签的授权、查询与成本分摊。
https://www.tencentcloud.com/document/product/1131/44513

[23] 腾讯云文档：CAM 概述。用户、组、角色、策略与资源权限。
https://www.tencentcloud.com/document/product/598/72360

[24] 腾讯云：CloudAudit 产品与文档。账户活动、API 使用、资源变更追踪、合规与故障排查。
https://www.tencentcloud.com/product

[25] 火山引擎文档：资源项目管理。项目分组以及对资源集合的 IAM 授权。
https://www.volcengine.com/docs/6257/170353?lang=en

[26] 火山引擎文档：Terraform。基础设施即代码提供程序设置与资源管理。
https://www.volcengine.com/docs/6706/101989?lang=en

[27] 火山引擎文档：采集 CloudTrail 日志。把云操作审计日志投递到 Torch Log Service。
https://www.volcengine.com/docs/6470/2201696?lang=en

第一篇 / 第 3 章

多云架构：责任共担、平台基因与工程意图

在不假装各云相同的前提下转译架构

多云架构是把工程意图严谨地转译到不同责任边界、资源模型、交付位置、集成模式和经济契约的过程。

本章内容

- 为何多云是架构与转译问题，而不是产品比较练习
- 随着服务托管程度提高，责任共担边界如何移动，同时不消除客户问责
- 为何责任必须涵盖可靠性、身份、数据、成本、证据和 AI 结果，而不只是安全
- 四维模型如何区分服务层、运行边界、交付位置和集成程度
- 平台基因如何解释路径依赖的产品侧重，同时避免落入技术决定论
- 为何使用 Azure 作为主要教学参考，以及阿里云、腾讯云和火山引擎如何体现不同优势
- 如何从资源、身份、网络、生命周期、证据、经济性和非等价性角度转译工程意图
- 为何必须分别评估功能、运维、治理、经济和组织等价性
- 哪些多云模式目标明确，哪些反模式只是复制复杂性
- 如何为一项云能力编写一页工程意图转译记录

多云架构经常以清单的方式介绍：两个公有云，也许再加一个私有云，以及一组看似相互对应的产品。这样的清单描述了采购与放置位置，却还没有描述架构。当团队能够说明为什么把能力放在某处、哪些属性必须保持不变、哪些责任转移给提供商、接受了哪些差异，以及如何把由此形成的系统作为一个统一工程资源环境治理时，架构才真正开始。

核心难点不在于不同云使用不同名称，而在于它们以不同方式打包责任。两个云中的托管数据库、Kubernetes 服务、模型终结点、搜索系统或智能体平台可能提供相似的应用功能，却公开不同的资源边界、身份模型、私网选项、升级行为、证据、限制和成本分摊方式。因此，名称到名称的映射在功能层面可能正确，在运行契约层面却可能错得非常危险。

本章使用三个概念让比较更加严谨。责任共担识别某一具体服务配置中仍由客户承担的义务；四维多云模型区分服务层、运行边界、交付位置和集成程度；平台基因则提供一种谨慎的编辑视角，用来理解提供商为什么反复强调某些集成模式。最终的转译单位是工程意图：对期望结果、约束、证据和可接受权衡作出的提供商中立表述。

本书把 Azure 作为主要参考平台，是因为它的管理层次结构和跨服务约定让控制平面概念在教学中格外清晰。阿里云、腾讯云和火山引擎并不被视为次要仿制品；它们是一等比较对象，其不同的产业历史与产品组合，揭示了哪些云抽象真正共有，又有哪些表面等价关系会破裂。

本章论点

多云架构不是同时使用多个提供商的产品目录，而是把工程意图严谨转译到不同责任边界、资源契约、运行假设、交付位置和集成模式的过程。



图 3-1 同一个工程意图可能需要四种不同的提供商专用资源图。

3.1 多云是一个转译问题

只有在架构具有明确目的后，“多云”一词才有意义。组织可能因为收购历史、区域可用性、监管、商业承诺、客户邻近性、专业数据或 AI 服务、韧性目标，或有意避免对单一提供商过度集中而使用多个云。这些动机不能互换。面向监管放置设计的架构，与面向价格套利或获取专业模型设计的架构，评估方式不同。

混合云与多云也描述系统的不同方面。混合强调跨越提供商区域、客户本地、专用环境或边缘位置等部署环境的连接；多云强调使用或治理一个以上提供商的服务。系统可以是混合云但不是多云，也可以是多云却没有深度互连，还可以兼而有之。关键问题不在于使用哪个标签，而在于哪些资源契约、数据路径、身份和责任跨越了边界。

可移植的应用制品并不会自动带来可移植的运行模式。一个容器可以运行在多种托管 Kubernetes 服务上，但集群身份、节点生命周期、CNI 集成、负载均衡器行为、存储类、升级通道、审计日志和成本计量项仍因提供商而异。OpenAI 兼容的推理请求可以访问多个模型平台，但部署容量、内容安全、数据驻留、速率限制、可观测性和退役策略仍可能不同。因此，不能笼统声称“可移植”，而必须拆解其含义。

```
多云决策 = 工程意图
         + 责任边界
         + 资源契约
         + 平台上下文
         + 已验证的非等价性
```

问题	薄弱的多云回答	架构层面的回答
为什么使用多个云？	避免供应商锁定	具有可度量验收标准的具名业务、监管、可靠性、能力或经济目标
哪些内容可移植？	应用	具体的制品、接口、架构、策略和操作规程，并分别说明其限制
不同提供商之间会改变什么？	只有产品名称	责任、资源粒度、身份、网络、生命周期、证据、配额和成本契约
如何验证成功？	部署已经完成	服务满足功能、安全、SLO、证据、成本与退役要求
退出策略是什么？	Terraform 可以重新创建它	数据、身份、状态、DNS、机密、证据、承诺和依赖资源都有记录完备的转换路径

可移植性具有多个维度

源代码可移植性、制品可移植性、API 兼容性、数据可移植性、运维可移植性、治理可移植性和商业可移植性是不同的主张。设计应明确自己作出的是哪一种主张。

3.2 责任共担是一条移动的边界

责任共担经常被呈现为一张安全图：提供商保护云本身，客户保护放入云中的内容。这一表述是有用的起点，但对架构而言过于狭窄。同样的边界移动也适用于可靠性、修补、扩缩容、备份、恢复、网络配置、身份、数据治理、可观测性、成本控制，以及 AI 平台上的模型选择、依据性、评估、工具权限和可接受结果。

随着提供商管理更多堆栈层，边界会向上移动。在 IaaS 中，提供商运维设施、硬件和虚拟化，客户通常控制来宾操作系统、中间件、工作负载、身份、网络规则和数据。应用 PaaS 把大部分操作系统与运行时负担转移给提供商，但应用配置、依赖项、数据、授

权、发布策略和 SLO 仍由客户负责。AI PaaS 可以管理模型服务、容量抽象、安全筛选器、检索基础设施、评估服务和智能体运行时，但客户仍须负责用例适用性、数据权利、访问边界、工具权限、评估标准和业务后果。Microsoft、阿里云、腾讯云和火山引擎都记录了这种边界在具体服务中的表现。[2] [3] [4] [9] [10] [15] [16] [17] [23] [24] [25]

责任还取决于配置。一个启用公网访问、客户管理密钥、跨区域副本和自定义备份策略的托管数据库，与同一服务采用私有访问、提供商托管加密和默认保留设置时相比，会产生不同的客户义务。托管 Kubernetes 服务可以运维控制平面，却仍把节点、附加组件、网络策略、工作负载、机密和数据保护留给客户负责。服务名称不能替代针对具体配置的责任审查。

服务抽象层级与责任边界

	IaaS	应用 PaaS	AI PaaS	SaaS / 智能体 SaaS
物理设施	提供商	提供商	提供商	提供商
硬件与虚拟化	提供商	提供商	提供商	提供商
操作系统与运行时	客户	提供商	提供商	提供商
平台中间件	客户	提供商	提供商	提供商
应用逻辑	客户	客户	客户	提供商
身份与配置	客户	客户	客户	客户
数据、策略与结果	客户	客户	客户	客户

实际边界取决于服务和配置；客户始终负责业务意图、数据分类和可接受的结果。

图 3-2 责任边界随服务抽象向上移动，但客户对意图和结果的问责仍然存在。

责任领域	IaaS 侧重点	应用 PaaS 侧重点	AI PaaS / 智能体平台侧重点
运行时与修补	客户运维来宾操作系统和大部分中间件	提供商运维操作系统/运行时；客户控制受支持版本、依赖项和应用配置	提供商运维推理或智能体运行时；客户控制部署、提示词、工具、知识、策略和发布
可靠性	客户设计实例、可用区、备份和恢复拓扑	提供商提供平台可用性机制；客户定义 SLO、扩缩容、依赖项、数据和恢复要求	提供商提供托管服务与平台恢复；客户验证质量、回退、速率限制、工具、知识和业务连续性行为

责任领域	IaaS 侧重点	应用 PaaS 侧重点	AI PaaS / 智能体平台侧重点
身份与网络	客户配置工作负载身份、防火墙、路由和暴露方式	提供商集成身份/网络功能；客户选择作用域、私有访问、入口和调用者	客户区分人员、工作负载、智能体、检索和工具身份，并约束每条数据路径
数据与合规	客户对数据进行分类、加密、备份和治理	提供商管理存储机制；客户负责数据权利、保留、架构和访问策略	客户负责训练/依据数据权利、检索授权、提示词/输出处理、记忆、驻留和监管解释
证据与成本	客户启用遥测、审计、标签、预算和优化	提供商发出平台遥测；客户把它与应用结果及成本分摊关联起来	客户关联模型、检索、工具、安全、评估、令牌、容量和业务结果证据

托管不等于无人负责

提供商可以自动化某项操作，却不会因此接手客户选择正确策略、验证结果、满足业务 SLO 或证明合规的责任。自动化改变的是谁来执行工作，并不会自动转移问责。

3.3 责任不止于安全

安全是人们最熟悉的责任共担领域，因为未授权访问会产生直接后果。生产架构至少还需要另外五本责任台账。可靠性台账说明谁负责设计冗余、检测故障、启动恢复并测试结果；数据台账说明谁控制分类、驻留、血缘、备份、删除和法律依据；运维台账说明谁负责修补、升级、观测、响应和退役；经济台账说明谁分摊成本、管理承诺、响应异常并证明节省；AI 台账说明谁选择模型与知识、评估行为、约束工具、处理有害输出并接受业务结果。

这些台账揭示了通用提供商/客户图所隐藏的缺口。例如，提供商可能保证托管搜索服务的可用性，而客户仍负责索引新鲜度、文档级授权、检索质量、别名发布和引用正确性。模型终结点可能满足基础设施 SLA，应用却会因为模型版本退役、配额耗尽、检索返回未授权证据，或工具超出预期作用域执行而失败。

多云设计应在选择“等价”产品之前建立这些台账。这项工作会迫使团队识别：哪些义务应在组织层统一实现，哪些可以委派给提供商原生控制，哪些在每个云中仍然属于应用专用责任。

台账	必须指定负责人的问题	典型证据
安全	谁授予权限、保护机密、约束网络、检测滥用并撤销访问？	角色分配、策略结果、防火墙规则、密钥历史、审计事件

台账	必须指定负责人的问题	典型证据
可靠性	谁定义 SLO、冗余、备份、恢复、故障转移、容量和测试?	架构决策、还原测试、故障转移记录、健康与错误预算消耗率信号
数据	谁对数据及衍生知识进行分类、定位、保留、删除和授权?	目录、血缘、保留策略、访问筛选器、删除与清除证据
运维	谁负责升级、变更、观测、响应、回滚和退役?	发布记录、遥测、事件、变更审批、退役资料包
经济性	谁负责分摊、预测、预算、承诺、优化和价值验证?	标签、账单导出、分摊规则、预算、承诺、已实现节省证据
AI 行为	谁批准模型、知识、提示词、工具、评估、安全边界和人工复核?	评估套件、安全结果、工具策略、追踪、审批记录、结果复核

3.4 云放置的四维模型

通常使用的公有/私有/混合/多云词汇，把多个决策混入同一个标签。本书采用四维模型，让这些决策可以分别作出。第一维是服务层：消费者获得的是基础设施、托管应用平台、软件，还是 AI 专用平台能力。第二维是运行边界：消费者是自行管理能力、与提供商共同管理，还是使用完全托管服务。第三维是交付位置：服务和数据在何处执行。第四维是集成程度：身份、网络、计费、策略、监控、开发和支持与周围云环境的集成深度。

该模型避免了一项重要的类别错误。在客户设施内交付的服务仍可由提供商运维；物理部署在另一个云中的数据库仍可保留原提供商的运行模式；容器平台可以运行在本地，却由公有云控制平面治理。反过来，同一提供商区域中的两个服务，如果使用彼此独立的身份、计费作用域、策略和运维证据，仍可能集成不良。

因此，每个工作负载都可以表示为四个维度中的一个点——或一条轨迹。迁移可能只改变交付位置，而保留服务层和运行边界；现代化改造可能从自主管理 IaaS 转向托管 PaaS，却不改变区域；多云集成可能提高身份与计费集成程度，同时仍由不同提供商分别负责其服务。



图 3-3 四维模型区分服务层、运行边界、交付位置和集成程度。

维度	代表性选项	主要设计问题	常见错误
服务层	IaaS、应用 PaaS、SaaS、AI PaaS	使用的是哪种能力和抽象?	把服务层当作成熟度排名，而不是责任选择
运行边界	自主管理、共同管理、完全托管	谁负责修补、扩缩容、备份、还原、升级和支持?	假定“托管”会转移所有运维职责
交付位置	提供商区域、另一个云、客户站点、主权环境、边缘	处理、数据、密钥和故障域位于何处?	把物理位置等同于运营方或控制平面
集成程度	身份、网络、计费、策略、遥测、开发者工作流、支持	独立服务应在多大程度上表现为统一云资源环境?	证据与治理仍然碎片化，却把已连接的服务称为已集成

独立使用各个维度

不要根据交付位置推断运行边界，也不要根据提供商品牌推断集成程度。应在架构决策记录中写明全部四个取值。

3.5 平台基因：路径依赖，但并非决定论

云平台并非从一张白纸开始设计。提供商会从其最初实现规模化的行业中继承技术资产、运行习惯、客户关系、开发者生态和商业激励。本书把这种累积的路径依赖称为“平台基因”。它是一种编辑视角，不是供应商分类，也不表示提供商只能服务最初的市场。

有用的因果链是：原始业务、主导工作负载、反复出现的工程问题、可复用技术资产、云产品侧重以及 AI 延伸。企业软件持续产生身份、可管理性、向后兼容性、混合运行和组织治理方面的压力；商业产生交易规模、商家、搜索、数据处理、弹性、促销峰值和生态

集成方面的压力；社交、游戏和实时媒体产生连接、低延迟、内容分发、互动服务与开发者分发方面的压力；内容与推荐平台则产生数据流水线、实验、排序、媒体处理和 AI 原生基础设施方面的压力。

如果把基因当作证据，它就会变得危险。提供商的起源无法证明当前 SLA、安全属性、价格优势、区域可用性或技术优越性。收购、监管、开源参与、客户构成和战略投资都会改变产品组合。正确做法是先形成关于可能侧重点的假设，再通过官方文档和受控实现验证当前资源契约。

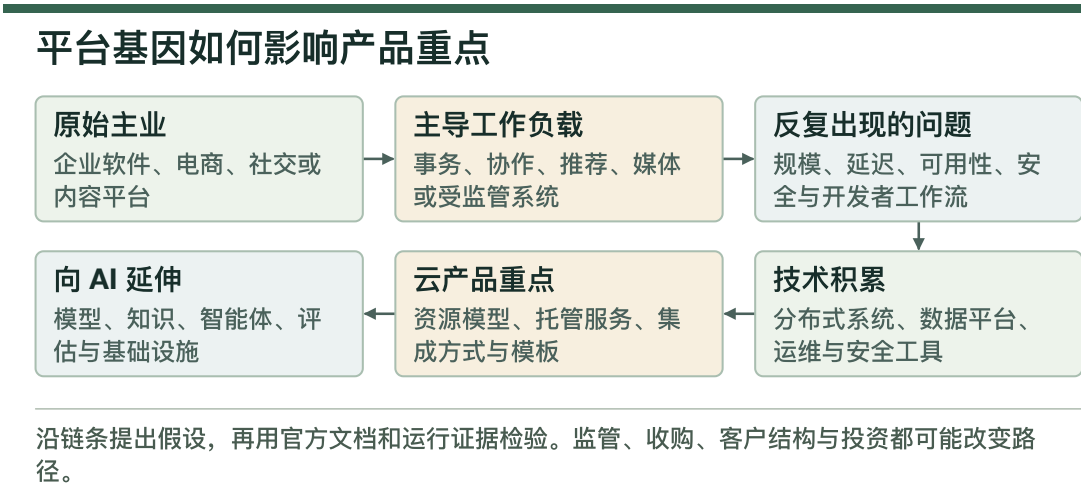


图 3-4 平台基因把原始业务压力与反复出现的云及 AI 平台侧重点连接起来，但每项结论都需要当前证据。

平台基因的安全用法	不安全用法
解释为什么某些产品集成、文档模式或解决方案模板格外显眼	声称提供商天生最适合与其原始业务相关的所有工作负载
为架构审查和比较测试生成问题	取代服务文档、基准数据、合同条款或区域验证
解释产品组合的一致性和可能的组织适配度	把历史起源当作不可改变的技术限制
识别周边生态和运行习惯造成的迁移摩擦	依据品牌声誉推断合规、主权、可用性或安全性
描述路径依赖，同时承认趋同与战略变化	把文化刻板印象或营销口号用作架构证据

3.6 四个云，四种可见侧重

以下概况有意保持简洁。它们描述有助于解释产品一致性和教学价值的可见侧重点，而不是排名。每项实现决策仍采用本章后文介绍的工程意图转译方法。

3.6.1 Azure：企业软件、开发者平台与混合一致性

Microsoft 进入云计算时，已经积累了数十年的企业软件、操作系统、数据库、生产力、身份和开发工具经验。其当前报告仍把 Azure、GitHub、SQL Server、Windows Server、Visual Studio、虚拟桌面和企业服务归入广义的智能云业务，而 Microsoft 365 及相关产品进一步加强了身份与组织集成。[8]

这段历史体现在 Azure Resource Manager、管理组、订阅、资源组、提供程序命名空间、Entra 身份、Azure Policy、Monitor、Cost Management、混合服务，以及对企业治理的一贯强调上。本书把 Azure 用作主要教学参考，是因为这些跨服务约定让资源层次结构、责任、身份、策略、证据和成本比较容易呈现为统一连贯的控制平面模型。这是一项编辑与教学判断，并非声称它在所有技术方面都普遍优越。[5] [6] [7]

3.6.2 阿里云：商业规模、交易、数据与生态运营

阿里巴巴把商业和云计算称为核心业务。其商业生态持续承受大规模交易、商家运营、搜索、物流协同、促销峰值、数据处理和合作伙伴集成的压力；其云产品组合通过弹性计算、存储、网络、安全、大数据和 AI 服务延伸了这些资产。[12]

这一路径体现在阿里云基础设施、数据、资源管理、编排和行业平台服务的广度中。资源目录、资源组、资源中心、标签、RAM、ROS、ActionTrail 以及托管数据与 AI 平台，提供了企业比较所需的控制平面锚点。架构层面的结论并不是商业工作负载必须使用阿里云，而是商业规模的运行经验有助于解释数据、交易、弹性和生态模式为何格外突出。[9] [10] [11]

3.6.3 腾讯云：连接、互动、游戏与实时媒体

腾讯的业务组合与社交网络、通信、游戏、视频、小程序、内容和数字服务紧密相关。这些工作负载持续要求低延迟互动、实时音视频、内容分发、大规模用户身份、生态分发，以及与消费和企业应用集成。腾讯 2025 年的报告仍把游戏、社交网络、视频号、小程序、云和 AI 描述为主要业务领域。[18]

因此，在连接、互动媒体、游戏、应用生态和实时服务很重要的场景中，应仔细考察腾讯云平台。其控制平面映射使用 CAM 管理权限，使用项目和标签进行组织，使用服务专用资源 API，并用 CloudAudit 留存管理证据，同时结合提供商专用的网络与数据服务。同样，平台基因视角只提出考察方向，当前服务契约才决定最终选择。[13] [14] [15] [16] [17]

3.6.4 火山引擎：内容、推荐、数据、媒体与 AI 原生基础设施

火山引擎是字节跳动的云与 AI 平台。字节跳动公开的产品组合包括 TikTok、抖音、CapCut、Lark、飞书和今日头条等全球与中国市场的内容、创作、协作和信息平台。这些系统持续产生推荐、实验、数据处理、媒体流水线、内容理解、增长分析和大规模 AI 方面的压力。[26] [27]

由此形成的侧重点体现在火山引擎的数据与 AI 定位、多媒体能力、豆包模型服务、方舟、AgentKit 和云原生 AI 基础设施中。其控制平面锚点包括项目、IAM、资源中心、标签、服务 API 和审计服务。该平台对本书尤其有价值，因为它展示了 AI 原生叙事如何源自互联网内容与推荐的发展脉络，而不是企业软件或商业的发展脉络。[19] [20] [21] [22] [23] [24] [25]

提供商	产品组合中可见的历史压力	需要检查的控制平面侧重点	没有证据时不得推断
Azure	企业软件、数据库、生产力、身份、开发工具、混合资源环境	分层资源治理、身份、策略、监控、成本、企业与混合集成	对每一种企业工作负载都最适合、成本最低或服务最强
阿里云	商业、交易、商家、物流、搜索、数据、促销规模弹性	弹性基础设施、数据平台、资源组织、编排、行业与 AI 服务	在所有商业或数据工作负载上自动胜出
腾讯云	社交连接、通信、游戏、视频、小程序、互动服务	身份与生态集成、实时媒体、连接、服务 API、内容与游戏能力	普遍低延迟，或与腾讯消费级平台直接等价
火山引擎	内容、推荐、实验、媒体处理、增长分析、AI	数据/AI 平台、多媒体、模型与智能体服务、项目、IAM、资源与审计控制	“AI 原生”意味着无需验证治理、生命周期、隔离或成本

3.7 工程意图是转译单位

产品不是多云转译的正确单位，工程意图才是。工程意图是对要实现的结果、要保留的约束、要收集的证据和组织愿意接受的权衡作出的提供商中立表述。“使用 Azure AI Search”是一项产品决策；“提供私有、按文档授权的混合检索，具备版本化摄取、引用、延迟目标、可审计查询和租户级成本分摊”则是一项工程意图。

意图必须比功能请求更精确。它应说明业务结果、数据分类、可用性与延迟目标、权限模型、放置约束、变更与退役要求、证据、成本负责人和可移植性预期。随后再把这些不变量转译为每个提供商的资源模型。转译结果可以选择单一托管资源、一组资源组合，也可以由客户自建控制措施弥补提供商功能缺失。

最终记录还必须说明非等价性。如果一个提供商原生支持文档级授权，另一个要求在应用代码中筛选，这就是架构差异。如果一个模型平台把专用吞吐量公开为部署资源，另一个通过账户配额管理容量，这一差异会影响隔离、证据、成本和运维。专业比较会保留这些差异，而不是把它们压扁成一个勾选框。

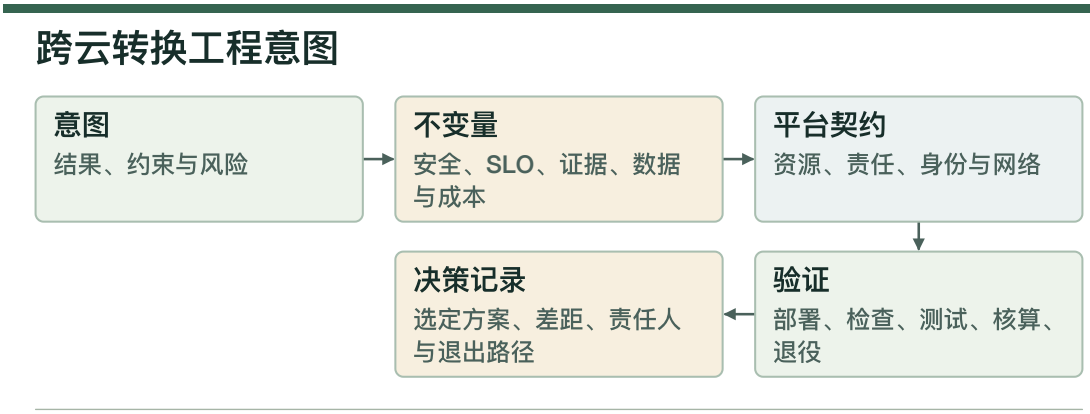


图 3-5 工程意图转译保留不变量、验证提供商契约并记录非等价性。

转译视角	问题	要收集的证据
资源模型	哪些具名资源、子对象、作用域和依赖项实现了该意图？	资源 ID、类型、层次结构、API/IaC 架构、依赖关系图
责任	在这一具体配置中，哪些职责属于提供商、客户或双方共同承担？	服务责任文档、操作规程、RACI
身份	存在哪些人员、工作负载、智能体、检索和工具主体，作用域分别是什么？	角色定义、分配、令牌/凭据、被拒绝操作测试
网络与放置位置	控制路径和数据路径在何处运行？数据、密钥和故障域位于何处？	拓扑、私有终结点/DNS、路由、区域、驻留、暴露测试
生命周期	如何创建、更新、升级、回滚、恢复和退役资源？	IaC、操作状态、变更计划、还原与清理证据
证据与 SLO	能否重建变更、请求、质量、可靠性和事件过程？	审计事件、日志、指标、追踪、评估、相关性 ID
经济性	计量、限制、承诺、预算、分摊和优化的对象是什么？	SKU/计量项、配额、账单导出、标签、预算、单位成本模型
非等价性	哪个不变量更弱、被外部化、成本更高或运维方式不同？	缺口登记册、客户补偿控制、已接受风险、负责人

3.8 等价性是多维的

两项服务很少能够被简单判断为等价或不等价。等价性是一种相对于目的定义的关系。对象存储在基本 PUT 和 GET 操作上可能功能等价，却在身份集成、事件投递、生命周期策略、不可变性、复制、一致性行为、私有网络、清单和成本方面不同。托管 Kubernetes 服务可以运行相同的 Deployment 清单，却在集群升级、节点责任、工作负载身份、网络、入口、存储、策略和可观测性方面不同。

AI 服务让这一区别更加明显。OpenAI 兼容 API 可以减少应用迁移工作，却不能证明治理等价。模型版本、内容安全、容量、配额、私有访问、数据处理、评估、追踪、区域可用性、退役通知和计费都可能不同。API 兼容性很有价值，但只是资源契约的一层。

等价性维度	含义	不匹配示例
接口	调用、架构、协议或制品可以复用	OpenAI 兼容请求可以工作，但流式事件或工具架构不同
功能	服务能够提供所需的应用能力	两者都能检索文档，但只有一个支持所需的混合排序或授权
运维	团队能以相近工作量进行部署、观测、恢复、升级和退役	Kubernetes API 相同，但节点、升级、入口和备份责任不同
治理	身份、网络、策略、审计、驻留和证据满足相同控制要求	一个服务支持私有终结点和细粒度角色，另一个需要公网访问或宽泛凭据
经济	计量、配额、承诺、分摊和单位成本的表现可以接受	实现等价吞吐量时，一个云需要专用容量，另一个采用可变的按请求付费
组织	平台符合技能、支持、采购、监管和生态约束	技术能力很强的服务，可能无法在组织现有模式下运维或采购

工程等价性始终相对于已声明的意图与证据集。
它不是两个产品名称之间的永久属性。

3.9 目标明确的多云模式

只有当放置模式服务于具名目标时，多云才有价值。架构应尽量减少运行时必须跨越的边界数量。跨云数据传输、身份联合、分布式追踪、事件协调、一致性和出口成本，都是真实的系统组件，而不是脚注。在每个提供商中复制每一层的设计，往往会使运维界面翻倍，却不能提供独立的故障保护或有意义的可选择性。

以下模式并不互斥。全球组织可以为区域系统使用提供商原生平台，为选定工作负载使用可移植容器基线，为清单和策略使用联合治理层，并使用另一个提供商的专业 AI 服务。关键是有意识地选择每种模式，而不是把同一种可移植性教条应用到整个资源环境。

模式	适用情形	主要收益	主要成本或风险
按领域采用提供商原生平台	某个提供商具有实质上更好的托管能力或生态适配度	最大限度利用托管服务和本地集成	迁移工作更多，并需要提供商专用技能
可移植工作负载基线	工作负载确实受益于通用制品和接口	跨环境可重复构建与部署	若过度扩展可移植性，会形成最低公分母设计
主提供商加恢复提供商	第二个提供商用于应对量化的集中风险或连续性风险	为选定业务能力提供独立恢复路径	数据复制、身份、运行手册、测试和闲置成本带来的复杂性
区域或监管放置	数据、客户、许可证或监管要求采用不同位置/提供商	满足本地合规、延迟或市场准入要求	若治理未联合，会导致运维与证据碎片化
专业服务组合	某项模型、数据、媒体、安全或行业服务具有独特价值	无需迁移整个资源环境即可使用差异化能力	跨云数据路径、出口流量、延迟、身份和合同耦合
联合控制与证据	资源保持提供商原生，但需要通用清单、策略意图、FinOps 和事件上下文	提供组织级可见性，同时不假装某一个提供商 API 拥有所有资源	如果丢弃源证据，标准化可能掩盖提供商专用语义

3.10 多云设计中的失败模式

最常见的多云失败始于简化。团队匹配产品名称、绘制对称方框、假定通用 API 会产生通用运维方式，或选择最低共享功能集。由此形成的架构在图中看似可移植，实际上会积累隐藏的提供商适配器、无人负责的责任、碎片化证据和不一致的安全控制。

第二类失败发生在平台本身成为抽象目标时。组织构建自定义层来隐藏每一项提供商差异，但该层随后必须复现身份、网络、策略、可观测性、配额、成本、故障行为和发布语义。组织实际上创建了一个新的云控制平面，却没有把它当作产品配备人员。只有当抽象能够消除重复复杂性，同时保留安全运维提供商原生资源所需的证据时，它才合理。

反模式	失败原因	修正方法
产品名称匹配	相似标签掩盖不同的作用域、生命周期、责任和证据	通过八个视角转译已声明的工程意图
对称架构图	视觉对称暗示错误的运行等价性	显示提供商专用资源图，并明确标记不对称性
最低公分母	为了保留假想的可移植性而舍弃托管能力	只为具有真实转换目标的制品与接口选择可移植性
把 API 兼容性当作证据	请求兼容并不能证明身份、安全、容量、遥测或退役等价	评估完整的资源契约与运行契约

反模式	失败原因	修正方法
用一个控制平面隐藏所有差异	抽象层会继承每个提供商的边界情况，并变成投入不足的平台产品	联合证据与意图；让提供商原生语义保持可见
无人负责的责任共担	每个团队都以为提供商或另一个团队会弥补缺口	维护针对具体服务和配置的责任台账
事后才考虑数据路径	出口流量、延迟、授权、驻留和一致性主导运行时行为	在选择服务之前设计跨云数据移动
把平台基因当作刻板印象	误把历史当作当前技术证据	只用平台基因生成问题；验证当前契约与证据

有益的不对称

优秀的多云架构可以有意保持不对称。不同提供商可以承担不同角色，因为工程意图、责任边界或平台适配度不同。对称本身并不是设计美德。

3.11 能力检查 C2：转译资源清单与所有权

这项能力检查生成一页工程意图转译，而不是完整部署。选择的意图有意保持基础性：发现某个组织作用域中的云资源，把每项资源与可问责的所有者和成本中心关联，并保留管理层次结构的证据。以后，治理、FinOps、安全审查和 CloudDesk 资源清单都可以使用同一能力。

该练习应使用官方文档和读者可用的每个提供商账户完成，但不要求实现全部四个提供商。先以 Azure 作为参考完成，再深入转译至少一个中国云，以识别一项真实的非等价性。其余提供商可以记录到资源契约层面。

工程意图字段	本次检查所需表述
结果	生成定义明确的组织作用域内的资源清单，包含提供商 ID、类型、位置、所有者、成本中心和管理父级
不变量	只读发现；不读取客户数据平面；最小权限；保留源资源 ID；报告未知所有权，而不是猜测
证据	作用域定义、身份/角色、清单查询或导出、标签/元数据结果、一条管理审计事件和时间戳
经济性	清单操作应免费或成本可忽略；不使用长期计算资源；注明账单/导出延迟
退役	移除临时身份、导出文件、脚本和测试标签；只保留决策记录与经过清理的证据
非等价性重点	层次结构、跨账户覆盖范围、标签强制执行、清单延迟、审计保留和成本分摊架构

提供商实现	资源与组织锚点	权限与证据	已知转译注意事项
Azure 参考	管理组 → 订阅 → 资源组 → ARM 资源；Resource Graph；标签	Entra 主体 + 作用域 RBAC；Activity Log；Cost Management 作用域	Resource Graph、策略、计费和 Activity Log 具有不同的新鲜度与保留语义
阿里云	资源目录/账户 → 资源组 → 服务资源；资源中心；标签	RAM 身份/策略；ActionTrail；账单/成本分摊服务	资源覆盖与跨账户聚合因服务和组织设置而异
腾讯云	组织/账户/项目或服务作用域 → 资源；项目与标签；服务清单	CAM 身份/策略；CloudAudit；计费数据	项目、标签和服务资源边界不一定与 Azure 资源组一致
火山引擎	账户/项目 → 服务资源；资源中心；标签	IAM 身份/策略；CloudTrail/审计服务；计费数据	必须逐项服务验证项目支持、资源中心覆盖范围和审计集成

```
intent_id: resource.inventory-and-ownership
scope:
  provider: azure
  management_scope: /subscriptions/<subscription-id>
principals:
  discovery_identity: <read-only-principal-id>
invariants:
  - no data-plane content access
  - preserve provider resource IDs
  - report missing owner or costCenter as a finding
translation:
  resource_model: management-group/subscription/resource-group/resource
  inventory: Azure Resource Graph
  ownership_metadata: tags.owner + tags.costCenter
  management_evidence: Azure Activity Log
non_equivalence:
  - tag policy and billing allocation are separate control-plane contracts
  - inventory and cost data have different freshness windows
verdict: acceptable-with-documented-gaps
```

验证需要的远不止资源列表截图。保存确切作用域、调用者身份、查询或导出定义、时间戳和至少一个资源 ID。确认调用者不会仅仅因为能够发现资源，就能读取受保护的数据平面对象。把缺少所有者或成本中心元数据的资源记录为发现项。最后，说明由哪项提供商原生控制预防或修复该缺口。

结果应控制在一页之内。它不是产品矩阵，而是一份决策记录，说明某项意图如何实现、哪些义务仍由客户承担，以及提供商差异会在何处影响治理或未来自动化。

验收标准

当审查者能够把意图追溯到具名提供商资源和作用域、复现证据、识别至少一项有意义的非等价性，并看到补偿控制由谁负责时，转译即通过验收。

CloudDesk 复用：提供商—意图转译

CloudDesk FinOps 智能体可以把本章复用为只读转译能力。智能体不应通过匹配名称来选择产品；它应接收工程意图和不变量，检查提供商资源契约，返回提供商专用实现，并保留非等价性登记册。任何后续执行能力都必须保持独立，并要求自己的权限、计划、审批和证据。

```
capability: cloud.capability.translate
risk_level: R0
mode: read_only
inputs:
  intent_id: string
  provider_targets: array
  required_invariants: array
  workload_context:
    data_classification: string
    regions: array
    slo: object
    cost_owner: string
outputs:
  provider_realizations: array
  responsibility_matrix: object
  resource_contracts: array
  evidence_sources: array
  non_equivalence_register: array
  recommended_pattern: string
  unresolved_questions: array
execution_constraints:
  - preserve provider-native identifiers and terminology
  - distinguish documented facts from architectural inference
  - never infer equivalence from product name or API compatibility
  - require human review before any provider-selection or change action
```

CloudDesk 用途	本章价值	必要的防护措施
FinOps 清单	跨提供商转译账户、项目、资源、标签、计费和证据作用域	不得在标准化过程中丢失提供商原生 ID 或计费语义
架构建议	比较运行边界、放置位置、集成程度和非等价性	返回选项与证据；不要声称存在普遍最优的云

CloudDesk 用途	本章价值	必要的防护措施
工具选择	把意图匹配到受治理的能力卡，而不是原始 API	必须明确身份、网络、生命周期、风险、成本和证据
迁移规划	区分可移植制品与提供商专用运行契约	在计划中纳入数据、身份、承诺、遥测与退役

本章小结

- 多云架构始于目的和工程意图，而不是提供商产品清单。
- 随着服务抽象上移，责任共担边界会移动，但客户仍须对身份、数据、策略、业务意图、成本和可接受结果负责。
- 应从安全、可靠性、数据、运维、经济性和 AI 行为等方面记录责任。
- 四维模型区分服务层、运行边界、交付位置和集成程度。
- 平台基因是谨慎的路径依赖视角：原始业务会产生反复出现的工作负载与技术资产，从而影响云和 AI 平台的侧重点。
- Azure 提供主要教学参考；阿里云、腾讯云和火山引擎体现同等重要却各不相同的产业与工程轨迹。
- 工程意图通过资源模型、责任、身份、网络、生命周期、证据、经济性和非等价性进行转译。
- 接口、功能、运维、治理、经济和组织等价性是彼此独立的主张。
- 目标明确的多云架构往往是不对称的，并会尽量减少不必要的跨云运行时边界。
- 一页转译记录比产品检查清单更有用，因为它保留提供商专用契约，并为缺口指定负责人。

参考资料与官方文档

产品名称、文档状态和引用的提供商描述核查于 2026 年 8 月 25 日。云产品、预览状态、区域可用性、配额、价格和责任边界都会变化；实施前请核验最新官方文档。

[1] 美国国家标准与技术研究院：《NIST 云计算定义》，SP 800-145。https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[2] Microsoft：云中的责任共担。https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility

[3] Microsoft：可靠性责任共担。https://learn.microsoft.com/en-us/azure/reliability/concept-shared-responsibility

[4] Microsoft：AI 工作负载责任共担。https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility-ai

[5] Microsoft：云采用框架概述。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/overview

[6] Microsoft：混合云与多云的统一运维。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/hybrid/strategy

[7] Microsoft：Azure Resource Manager 概述。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview

[8] Microsoft：2025 年年度报告。https://www.microsoft.com/investor/reports/ar25/index.html

[9] 阿里云：ECS 责任共担模型。https://www.alibabacloud.com/help/en/ecs/user-guide/ecs-shared-responsibility-model

[10] 阿里云：ACK 责任共担模型。https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/security-and-compliance/shared-responsibility-model

[11] 阿里云：资源管理概述。https://www.alibabacloud.com/help/en/resource-management/

[12] 阿里巴巴集团：2024 财年年度报告。
https://static.alibabagroup.com/reports/fy2024/ar/ebook/EN/index.html

[13] 腾讯云：访问管理概述与资源描述。https://www.tencentcloud.com/document/product/598/10583 and https://cloud.tencent.com/document/product/598/10606

[14] 腾讯云：标签。https://cloud.tencent.com/document/product/651

[15] 腾讯云：数据处理与安全协议。https://www.tencentcloud.com/document/product/301/17347

[16] 腾讯云：TRTC 安全与责任共担。https://www.tencentcloud.com/document/product/647/53568

[17] 腾讯云：专属云中心责任共担。https://www.tencentcloud.com/document/product/1226/61856

[18] 腾讯：2025 年中期报告与业绩资料。
https://static.www.tencent.com/uploads/2025/08/13/56af6c5c98acdc7bd757a7bd208d8189.pdf

[19] 火山引擎：身份与访问管理。https://www.volcengine.com/docs/6257/64959

[20] 火山引擎：资源中心。https://www.volcengine.com/docs/6649/1109337

[21] 火山引擎：项目。https://www.volcengine.com/docs/6649/94337

[22] 火山引擎：标签。https://www.volcengine.com/docs/6649/189377

[23] 火山引擎：云服务责任共担条款。https://www.volcengine.com/docs/6256/79758

[24] 火山引擎：VKE 责任共担。https://www.volcengine.com/docs/6460/2179469

[25] 火山引擎：AgentKit 责任共担。https://docs.volcengine.com/docs/86681/2605800?lang=zh

[26] 字节跳动：公司与产品组合。https://www.bytedance.com/en/

[27] 火山引擎：云与 AI 平台概述。https://www.volcengine.com/

基础云资源控制平面

应用上线前，资源放在哪里，谁能改动它，网络怎样到达它，数据能否恢复？这五章从资源组织与权限开始，依次检查计算、网络与存储，把部署意图落实到可验证的依赖图。

04 资源组织、标签与基础设施即代码

05 身份与最小权限

06 计算、弹性与可靠性

07 网络、私有连接与暴露控制

08 存储、数据湖与事件驱动的数据路径

第二篇 / 第 4 章

资源组织、标签与基础设施即代码

作用域、元数据和声明状态如何让云资源环境可归属、可复现、可审查、可销毁

云资源不会仅仅因为出现在门户中就变得可管理。只有当它们位于明确作用域、携带受控元数据、能够作为清单查询，并由可审查的期望状态生成且具有经过验证的退役路径时，才变得可治理。

本章内容

- 为何资源组织是一份控制平面契约，而不只是整理门户界面
- 组织、账户、订阅、资源组、项目和单个资源如何形成不同的策略、计费与生命周期边界
- 为何不能把提供商资源 ID、人类可读名称和标签视为可互换的标识符
- 如何为所有权、环境、成本分摊、数据分类、关键性和生命周期设计标签词汇
- 为何清单必须支持跨作用域查询，并与活动、合规和成本证据关联
- 基础设施即代码如何把云配置变成版本化期望状态、计划变更和确定性退役
- Bicep 或 ROS 模板等原生声明何时不同于 Terraform 等基于提供程序的工作流
- 状态、漂移、导入、重构、模块、提供程序版本和机密如何影响 IaC 安全
- 如何从验证、计划到应用、核验和销毁，建立能够产生证据的变更循环
- 两个相互独立的 Azure 实验如何建立资源归属和完整 IaC 生命周期，并简要映射到阿里云、腾讯云和火山引擎
- CloudDesk FinOps 智能体以后如何复用组织、标签审计、部署和差异能力，而本章不予实现

云资源环境往往还没变大，就已经变得混乱。几个实验以个人名称创建；一个共享数据库被放进不属于任何应用的资源组；生产 API 带有所有者标签，但其存储和网络资源没有；项目结束后，一个手工创建的资源仍然存活，因为没人知道能否安全删除；后来的迁移引入了 Terraform，但现有资源仍处于其状态之外。表面症状是杂乱，真正的架构问题则是所有权、生命周期、证据和成本从未得到明确表达。

因此，资源组织不是“真正”架构之前的装饰性序曲，而是可编程云控制平面的第一项纪律。作用域决定身份在何处获得权限、策略在何处继承、配额和预算在何处评估、变更如何隔离，以及删除的影响范围在哪里受限。名称帮助人员导航；标签附加可变的业务含义；资源 ID 提供权威地址；清单服务让生成的资源图可供查询。随后，基础设施即代码把期望状态、依赖关系和生命周期操作纳入可审查的工程工作流。

本书背后的课程材料用实践语言描述了从 ClickOps 到 IaC 的转变：门户创建是一次性配置操作，而 IaC 可版本管理、可审计、可复现、可销毁。本章把这项观察发展为资源级契约。它不教授 Bicep 或 Terraform 的每项语言特性，而是说明组织与声明如何协同，使云资源环境在任何时刻都能回答四个问题：存在什么？谁拥有它？它应处于什么状态？如何安全变更或移除？

Azure 通过资源组、标签、Azure Resource Graph、Cost Management、Bicep 与 Resource Manager、部署历史、What-If、部署堆栈和 Terraform，提供完整实现路径。阿里云、腾讯云和火山引擎则按工程意图进行比较。它们的组织结构、标签系统、清单服务、编排产品和 Terraform 提供程序在产品边界上有所不同，但都在解决同一个反复出现的问题：把不断增长的提供商资源集合转变为可归属、受控制的系统。

本章论点

资源组织与基础设施即代码是首要的控制平面纪律。组织为每项资源赋予作用域、所有者、用途和成本上下文；IaC 则为资源赋予可审查的期望状态、变更历史、依赖关系图和退役路径。



图 4-1 资源组织把分层权限与横向元数据、清单、策略和成本视图结合起来。

4.1 组织是一份控制平面契约，而不是整理文件夹

文件夹树回答项目显示在哪里；控制平面作用域则回答哪些身份可以操作、哪些策略适用、哪些成本会聚合、哪些配额约束工作负载、收集哪些证据，以及哪些资源可以一起变更。这些问题可能在门户层次结构中重合，但不是同一个问题。美观的文件夹布局仍可能掩盖薄弱的所有权和不安全的生命周期边界。

资源组织始于识别管理单元。组织或租户层协调多个账户；账户或订阅通常是计费、配额、服务启用和管理委派的强边界；资源组或项目是更小的操作作用域，可以汇集某个工作负载、环境、团队或生命周期的资源；资源仍是提供商可寻址的对象，数据平面和控制平面操作最终作用于此；标签与查询则建立横跨这一层次结构的视图。

优秀的组织模型会使常见操作局部化。平台团队应能向工作负载团队授权，而不授予无关访问；策略应能面向预期环境，而不必枚举数千个资源 ID；成本视图应能归属大部分直接支出，而无需手工维护电子表格；部署应能创建和移除一组内聚资源；事件响应人员应知道受影响的所有者与服务。若这些操作需要临时解释，组织模型就过于薄弱。

问题	控制平面答案	缺失时的失败
管理边界是什么?	组织、账户/订阅、资源组/项目、资源 ID	权限和策略适用范围过宽或不可预测
谁拥有该资源?	问责标签、工作负载目录、已分配的组或项目	事件与成本没有负责团队
它为什么存在?	用途、工作负载、环境、数据类别和生命周期元数据	孤立资源看起来仍然合理
它应处于什么状态?	版本化 IaC、参数、策略和提供程序版本	门户状态成为唯一规范
如何批准变更?	计划/What-If、策略检查、审查、部署身份、活动证据	无法重建配置变更
如何退役?	销毁计划、依赖项审查、保留规则、清理验证	可计费且涉及安全的残留对象继续存在

设计规则

选择作用域时，应确保权限、生命周期和成本容易理解。不要仅仅因为某个资源组存在就使用它，也不要用标签模仿本应通过账户、订阅、项目、策略或专用资源实现的边界。

4.2 作用域层次：组织、账户、资源组与资源

每个主要云都提供不止一层资源组织，但名称和语义各不相同。Azure 使用管理组、订阅、资源组和资源；阿里云结合资源目录、账户、资源组和标签；腾讯云企业组织协调成员账户与部门，项目和标签能力则在账户内组织资源；火山引擎提供项目、标签、资源中心和多账户资源管理服务。词汇相似并不保证生命周期或授权行为等价。

组织级层次结构应表达持久的公司结构和控制委派，不适合短期应用部署。账户或订阅是更强的缩放单元，可以分离生产与实验，隔离计费或配额，并限制管理员账户失陷的影响。资源组或项目更适合工作负载或环境，其资源可以一起运维和退役。单个资源仍保留服务专用生命周期与数据平面行为。

正确边界取决于失败后果。如果策略错误绝不能影响生产，应在账户或订阅层面分离生产。如果两项资源必须一起部署、观测和删除，应把它们放在同一部署边界，除非共享服务架构另有要求。如果一个数据库由十个应用共享，不要假装它完全属于某个应用资源组；应把它建模为共享平台资源，并明确消费者与分摊规则。

作用域	主要目的	典型决策	不当用法
组织 / 租户层次结构	多账户治理与委派管理	企业策略、身份中心、账户创建、合并计费	单次部署生命周期
账户 / 订阅	强管理、计费、配额与影响范围边界	生产隔离、法律实体、区域策略、平台所有权	为每个小组件单独建立账户
资源组 / 项目	操作分组与委派的工作负载管理	应用/环境所有权、策略作用域、成本范围、清理边界	用一个杂项组容纳整个公司
资源	提供商可寻址的托管对象	配置、数据平面访问、SLO、保留、替换	把名称当作唯一持久身份
标签 / 标记视图	横向分类与分摊	所有者、工作负载、环境、成本中心、数据类别、生命周期	把元数据当作强隔离边界

分组模式	适用情形	权衡
每个工作负载和环境使用一个资源组	资源共享所有者、发布节奏、证据和退役日期	共享服务需要独立分组和明确关系
每种环境类别使用一个订阅/账户	生产需要更强的策略、配额、身份和计费隔离	需要更多跨作用域网络、身份和运维协调
共享平台资源组	网络中心、注册表、可观测性或数据服务为多个工作负载服务	成本分摊与变更治理必须识别消费者
临时实验组	所有资源都有意保持临时状态，且必须一起删除	只有在验证清理且未放入持久数据时才有用
每组一个资源	仅在特殊策略或删除边界下偶尔合理	会产生管理噪声，削弱分组含义

4.3 名称、资源 ID 与标签承载不同语义

云工程师经常让名称承担过多职责：希望资源名称既标识提供商对象，又描述工作负载、编码环境、表明所有者、传达区域并支持成本分摊。它无法可靠地同时完成这些任务。名称受服务专用字符集、长度、全局唯一性和不可变规则约束；有些名称出现在数据平面终结点中，另一些可以在不改变提供商身份的情况下更改。命名约定应支持导航，但不能成为业务含义的唯一来源。

提供商资源 ID 是权威地址。它连接 API 操作、活动日志、策略状态、配置历史、成本记录和部署输出。资源可以在目录中改名而保留提供商 ID，也可以被新对象替代并复用人类可读名称。因此，证据系统应保留 ID、账户/订阅作用域、资源类型、区域和提供商，而不只是显示名称。

标签是可变的键值元数据。它们可以跨类型和区域对资源分类、驱动清单筛选、支持成本分摊并为自动化提供输入，因此很有价值；其灵活性也是弱点。不受控制的拼写、大小写、缩写、个人姓名和自由文本会形成无法一致查询的分类体系。标签以纯文本存储，可能出现在成本导出、部署历史、命令和日志中，绝不能包含凭据、个人数据或机密业务文本。

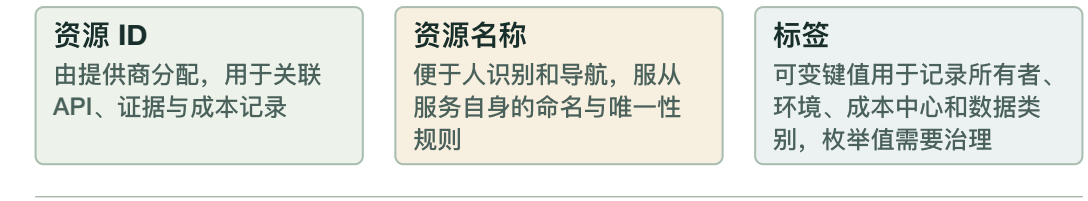


图 4-2 资源 ID、名称和标签职责不同，应进行关联而不是相互替代。

机制	最佳用途	稳定性	安全说明
资源 ID	权威 API、证据与计费关联键	在资源生命周期内保持持久	可能暴露账户、作用域、类型和路径，但仍不属于机密
资源名称	供人员导航与运维识别	因服务而异；重命名支持并不一致	不要编码个人或机密信息
标签键/值	可变业务分类与自动化输入	应随所有权和生命周期变化	纯文本元数据；不要存储机密
目录条目 / CMDB ID	跨提供商的企业工作负载、服务或产品身份	应比单个提供商资源存在更久	必须保留提供商 ID 与证据链接
IaC 地址	模块与状态内部的逻辑地址	只有在妥善管理重构时才稳定	更改地址可能导致替换，除非使用移动/导入语义

命名原则

名称只放置人员识别所需的属性；可变业务分类放在受控标签中；权威身份放在提供商资源 ID 与服务目录中。不要把资源名称当作数据库记录解析。

4.4 设计标签与成本分摊策略

标签策略是一个小型数据模型。在应用标签之前，必须定义键、值域、所有者、强制方式和消费者。策略应从团队需要作出的决策出发：谁响应事件、成本归属哪个产品、资源是否为生产资源、适用哪些数据处理规则、何时审查或删除资源，以及自动化能否修改资源。没有消费者的标签只是装饰性元数据。

最有用的标签可分为几类。问责标签识别所有者和成本中心；工作负载标签把资源关联到应用或产品；运维标签记录环境、关键性、生命周期和服务级别；治理标签对数据和合规上下文分类；自动化标签说明资源由 IaC、平台团队还是其他控制器管理。每个键都应有受控词汇和有文档说明的记录系统。

成本分摊并不只是附加标签。有些费用不能直接映射到资源，有些服务不会把所有资源标签写入使用记录，共享平台还需要分摊规则。标签通常只适用于添加后的使用量，不会重写历史计费记录。Azure Cost Management 可以按资源组或标签分组，并可对使用记录应用受支持的标签继承。中国云平台也有类似的分摊标签能力。因此，成熟设计会结合账户层次、资源作用域、标签、账单导出和派生分摊元数据。

FinOps 指南把分摊定义为：使用账户、标签、标记和其他元数据，把成本与使用量分配给负责的所有者、产品或成本中心。这一定义很重要：标签是分摊的输入，而不是分摊本身。标签合规率既应按资源数量衡量，也应按成本覆盖率衡量。一个未标记的共享集群可能比数百个已标记的测试对象更重要。

类别	建议键	示例值	主要消费者
问责	owner	team-platform; team-search; course-lab	事件响应、访问审查、孤立资源检测
财务	costCenter	cc-4102; research-ai; training	成本分析、成本展示、成本分摊
工作负载	workload	catalog-api; finops-agent; course-rag	服务目录、依赖关系映射、成本/单位经济性
环境	environment	dev; test; staging; prod; lab	策略、部署工作流、警报路由
用途	purpose	web-api; knowledge-index; backup; experiment	清单说明与优化
关键性	criticality	tier-0; tier-1; tier-2; noncritical	SLO、备份、恢复、变更审批
数据治理	dataClass	public; internal; confidential; restricted	安全、保留、驻留、审查

类别	建议键	示例值	主要消费者
生命周期	lifecycle	ephemeral; persistent; review-2026q4; retire	清理、保留、技术债务
自动化	managedBy	bicep; terraform; ros; platform-team	漂移处理与变更路由

标签反模式	失败原因	更好的规则
所有者使用个人电子邮件地址	人员会离职，而且个人数据会扩散到导出和日志中	使用持久的团队或服务所有权 ID
环境值同时包含 prod、production、PRD 和 live	查询与策略分配会遗漏资源	使用受控枚举
通过解析资源名称推导成本中心	重命名和服务命名限制会破坏分摊	显式存储并验证成本中心
所有标签均为可选	覆盖率会下降，孤立资源会累积	定义一小组必需标签并衡量合规性
设置数百个假想标签	维护成本高，却没有消费者	只有当某个流程会读取时才创建标签
标签中包含敏感值	标签会出现在多种运维和计费界面中	使用机密存储或受保护目录
默认把标签当作授权	元数据本身并不强制访问控制	只在明确支持并经过测试时使用 IAM/ABAC
标签应用一次后从不复核	所有权、环境和生命周期会过时	定期与记录系统进行对账

```
# Example organization tag contract (conceptual)
required:
  owner:       service-catalog.team-id
  workload:    service-catalog.workload-id
  environment: [dev, test, staging, prod, lab]
  costCenter:  finance.cost-center-id
  lifecycle:   [ephemeral, persistent, review, retire]
optional:
  dataClass:   [public, internal, confidential, restricted]
  criticality: [tier-0, tier-1, tier-2, noncritical]
  managedBy:   [bicep, terraform, ros, platform-team]
prohibited:
  - passwords, tokens, connection strings
  - personal or regulated data
  - free-form incident notes
```

分摊指标

同时衡量资源覆盖率和成本覆盖率。一个有用的成本加权指标是：合规直接成本除以可直接归属成本总额。共享与未分摊费用需要单独的分摊策略，而不是虚构标签。

4.5 清单必须可查询

门户中的资源列表还不是清单。清单是可查询的控制平面视图，能够跨资源类型、区域、账户和标签回答问题。它应返回提供商 ID、类型、位置、所有者、生命周期元数据和管理状态，也应暴露缺口：没有所有者的资源、位于未经授权区域的生产对象、声称由 Terraform 管理却不在预期状态中的对象，以及已超过删除日期的资源。

Azure Resource Graph 使用跨作用域资源元数据查询扩展了 Resource Manager。阿里云、腾讯云和火山引擎也提供类似的资源中心、配置和标签搜索服务。这些系统针对探索与治理优化，但不能替代活动日志、配置时间线、遥测或计费账本。清单回答当前可见什么；活动证据回答谁进行了变更；成本数据回答产生了什么费用；可观测性回答运行表现如何。

有用的清单查询会保留资源 ID，并投影标准化元数据，而不只是统计对象数量。查询应可导出为证据包，并可在自动化中重复执行。只要作用域与权限明确，同一个查询就可以服务于本书实验、治理审查或只读智能体工具。

```
Resources
| where resourceGroup =~ 'csaip-ch04-lab-rg'
| project
    id,
    name,
    type,
    location,
    owner = tostring(tags.owner),
    environment = tostring(tags.environment),
    purpose = tostring(tags.purpose),
    costCenter = tostring(tags.costCenter),
    lifecycle = tostring(tags.lifecycle),
    managedBy = tostring(tags.managedBy)
| order by type asc
```

证据来源	所回答的问题	不得推断
资源清单 / 资源图	此作用域当前可见哪些对象与元数据?	谁进行了每项变更，或服务是否健康
活动 / 审计日志	哪个主体在何时执行了控制平面操作?	完整的数据平面影响或业务原因

证据来源	所回答的问题	不得推断
配置时间线 / IaC 状态	属性与声明状态如何变化?	运行中的服务是否满足 SLO
指标、日志与追踪	工作负载在某个时间区间内表现如何?	权威计费或所有权
成本与使用数据	记录了哪些费用，如何分摊?	实时支出或完整标签覆盖
服务目录 / CMDB	应适用哪个产品、所有者、依赖关系和业务上下文?	提供商配置实际合规

4.6 基础设施即代码是一份托管变更契约

基础设施即代码经常被简化为“编写 YAML 或 HCL，而不是点击”。这一定义忽略了工程价值。IaC 是声明意图、提供程序架构、执行身份、当前基础设施状态、依赖关系图，以及预期与实际变更证据之间的一份托管变更契约。源文件很重要，因为它可以版本管理和审查，但文件本身并不是系统。

声明式 IaC 描述期望结果，并由引擎推导操作图；命令式脚本则描述一系列命令。两者都可以自动化，也都可能失败。声明式工具的价值在于能够比较期望状态与已知或观测状态、从引用构建依赖关系、预览变更，并让重复执行向同一结果收敛。幂等性是一项目标，而不是保证：提供商默认值、异步 API、非确定值、外部控制器和人工变更仍可能带来意外。

IaC 会产生多种持久工程制品。配置记录预期结构；计划或 What-If 记录预测变更；应用或部署记录标识执行过程与结果；输出向下游系统公开资源 ID 与终结点；状态或部署元数据把逻辑地址与提供商对象连接；验证记录测试实际结果；销毁计划与清理证明关闭生命周期。这些制品组合起来，比“门户刀片显示成功”的截图强得多。

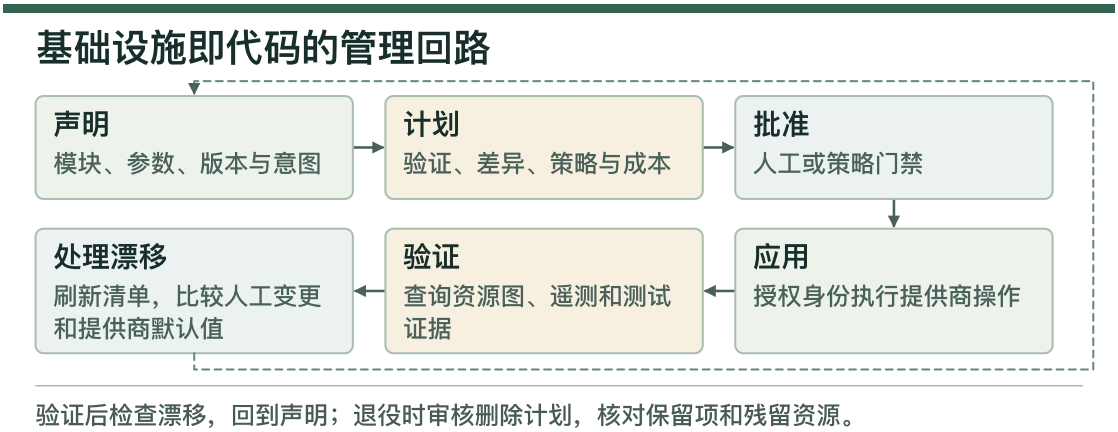


图 4-3 当声明、授权、证据、验证、漂移处理和退役相连接时，IaC 成为控制平面反馈循环。

属性	工程含义	证据
可版本管理	期望状态变更可以审查并与提交关联	存储库历史与发布记录
可复现	在有文档记录的提供商约束内，相同输入可以创建等价环境	参数、提供程序锁、部署输出、验证
可审查	执行前可以看到建议的资源图和破坏性操作	计划或 What-If 加策略结果
可审计	执行身份、时间、作用域和结果均有记录	部署历史、活动日志、流水线记录
可组合	模块公开类型化输入与输出，并保留依赖关系	模块接口与资源图
可感知漂移	能够检测并分类声明状态、已知状态与实际状态之间的差异	刷新、清单查询、漂移报告
可销毁	临时基础设施具有明确且经过验证的销毁路径	销毁计划、应用记录、零资源验证

4.7 原生 IaC 与可移植 IaC

云原生声明语言与单一提供商控制平面紧密对齐。Azure Resource Manager 模板和 Bicep 直接公开 Azure 资源类型与部署作用域。Bicep 编译为 ARM JSON，并使用 Resource Manager 部署。阿里云资源编排服务接受 ROS 模板，也提供托管 Terraform 执行。原生系统通常较早公开新的平台功能并产生提供商原生部署历史，但它们有意不向其他云移植。

Terraform 在不同提供商之间提供通用配置语言、计划/应用工作流、模块系统和状态模型。提供程序插件把 Terraform 资源操作转译为各云 API。它为团队提供可移植的工程工作流，而不是可移植的基础设施语义。Azure 资源组、阿里云资源组、腾讯云项目和火山引擎项目不会因为都用 HCL 声明就变得等价。模块仍必须编码提供商专用的身份、网络、生命周期和证据契约。

选择不应意识形态化。当架构专属于某个提供商、平台功能的一等支持很重要，或提供商部署模型应成为主要证据来源时，使用原生 IaC。当团队需要跨多个提供商或 SaaS 系统的一致工作流、已有成熟模块，或独立状态与规划模型更符合运行流程时，使用 Terraform。许多组织会同时使用两者：登陆区或强提供商专用服务采用原生声明，跨提供商平台组合采用 Terraform。

Azure 部署堆栈在传统部署记录之上增加了生命周期语义。堆栈把一组资源作为整体管理，并可根据取消管理行为，删除或分离从声明中移除的资源。这一点很重要，因为增量模板部署不会删除后续模板中省略的所有资源，而且删除场景下的完整部署模式正逐步由部署堆栈取代。必须有意选择删除语义，而不能根据“声明式”一词自行假定。



图 4-4 原生 IaC 与基于提供程序的 IaC 共享控制平面目标，但把状态、架构与生命周期责任放在不同位置。

维度	Bicep / 原生模板	Terraform / 提供程序工作流
主要架构	提供商原生资源类型与 API 版本	Terraform 提供程序资源架构
状态模型	实际资源状态保留在云控制平面中；部署记录操作	状态文件把逻辑地址映射到提供商对象与元数据
变更预览	ARM What-If 或原生堆栈预览	terraform plan，可选择保存为应用制品
功能提供时机	通常较早公开提供商功能	取决于提供程序发布与架构实现
跨提供商工作流	不是目标	通用语言和工作流，资源仍为提供商专用
删除模型	增量部署、部署堆栈或显式删除	配置/状态比较与销毁计划
运维风险	模板/API 版本漂移与提供商部署语义	状态安全、锁、提供程序升级、地址重构
最佳适用场景	深度提供商集成与原生控制平面治理	跨多个提供商与服务的一致编排

4.8 状态、漂移、导入与重构

Terraform 状态不是云的备份，而是 Terraform 在配置地址与远程对象之间使用的映射，还包含高效规划所需的元数据和缓存属性。状态丢失或泄露会破坏生命周期控制或暴露敏感值。因此，团队使用时需要受保护的远程后端、访问控制、加密、版本管理和锁。状态

绝不能提交到普通源码存储库。

Bicep 不要求作者维护等价状态文件。Resource Manager 在计算 What-If 或执行部署时观测当前 Azure 资源与部署元数据。这消除了一类状态操作，但没有消除漂移。门户编辑、策略修正、提供商默认值或外部控制器仍会改变实际状态。差别只在于映射与比较责任位于何处。

漂移是声明状态、工具已知状态与提供商实际状态之间任何实质差异。漂移并不全是恶意或意外造成的：事件响应人员可能进行紧急变更；自动扩缩容和托管服务会合理地改变运行状态；生命周期策略可能由另一个控制器管理。响应应先分类漂移：从代码对账、导入或记录、把属性委派给另一个控制器，或显式忽略。盲目强制每项差异回到配置可能造成中断。

现有资源可以纳入 IaC，但导入并不等同于自动完成架构工作。必须声明、导入或引用资源，比较并规范化，同时避免破坏性替换。模块重构时逻辑地址也可能变化。Terraform moved 块以及提供商原生导入或现有资源引用，有助于保留身份。重构后的计划如果建议替换，说明地址语义在没有迁移记录的情况下被改变，应引起警惕。

情形	首选响应	要保留的证据
未经授权的门户编辑	通过 IaC 还原，并限制直接写权限	活动事件、计划、审批、应用、验证
事件期间的紧急变更	先稳定系统，再通过经过审查的后续工作对账代码/状态	事件记录、资源差异、最终配置
托管服务或自动扩缩容器变更	适当时把运行字段排除在 IaC 所有权之外	控制器契约与忽略属性的理由
在 IaC 之外创建的资源	决定导入、引用、替换或退役	所有权决策与非破坏性计划
模块地址已更改	使用显式移动/重构语义	移动映射，以及显示无替换的计划
资源已从声明中移除	审查保留、分离、删除与清除要求	销毁/取消管理计划与数据保留审批

状态安全

把 IaC 状态存储视为敏感运维数据。它可能包含标识符、拓扑、计算属性和提供商返回的值。“机密不在 .tf 文件中”并不能证明状态中不存在机密。

4.9 模块、参数、环境与机密

模块应遵循所有权与生命周期边界，而不是供应商产品目录类别。网络模块可以复用，但完整的工作负载模块应连接该工作负载所需的网络、身份、存储、诊断、策略和成本元数据。“每种资源类型一个模块”的设计，只是在另一层薄封装背后隐藏提供商架构；覆盖整个企业的巨型模块则导致相反问题：每次变更都产生巨大的计划与影响范围。

环境差异应通过显式参数表达，必要时还要采用更强的作用域边界。生产与开发不应只因一个变量设置 environment=prod 而有所不同。生产可能需要独立订阅或账户、不同身份、私有终结点、更强策略、独立状态、受保护审批以及更长的保留路径。参数用于配置契约，本身不会建立隔离。

提供程序与模块版本属于期望状态。固定兼容范围，在工具支持时提交依赖锁，并把升级作为基础设施变更进行审查。提供程序升级可能改变默认值、验证、差异行为和资源替换。因此，版本升级后的计划不是文书输出，而是迁移评估。

机密需要独立路径。源码存储库、变量文件、计划制品、状态文件、部署历史、标签和命令日志都可能泄露值。优先使用工作负载身份与托管机密存储引用。当提供商要求把机密作为输入时，应明确值会出现在哪里、限制执行环境、把输出标记为敏感，并验证状态/后端保护。控制台输出中的遮盖并不是完整的机密管理策略。

```
cloud-infrastructure/
├── README.md
├── modules/
│   ├── workload-foundation/
│   ├── private-storage/
│   └── observability-baseline/
├── environments/
│   ├── dev/
│   │   ├── main.tf
│   │   └── dev.tfvars.example
│   └── prod/
│       ├── main.tf
│       └── prod.tfvars.example
├── policies/
├── tests/
├── evidence/
└── .gitignore
```

存储库制品	用途	发布规则
配置与模块	期望状态与可复用契约	版本管理、审查、代码检查、测试
示例变量文件	记录不含凭据的必需输入	提交示例；排除真实机密与状态
提供程序锁文件	使提供程序选择可复现	提交并审查升级
计划制品	记录建议的操作图	为获批运行保存；视为可能含敏感信息
状态后端	把配置地址映射到托管对象	加密、锁定、限制、版本管理；绝不公开
证据包	连接计划、应用、清单、活动与验证	按变更和审计策略保留

存储库制品	用途	发布规则
清理脚本或销毁计划	关闭临时环境生命周期	把它作为验收的一部分进行测试，而不是等本书发布后再测试

4.10 IaC 变更循环：计划、审批、应用、验证、退役

成熟的 IaC 工作流会把预测与权限分开。格式化、静态验证和模块测试证明声明内部一致；计划或 What-If 预测提供商操作；策略检查判断建议状态是否允许；成本分析识别重要经济影响；人工或自动审批授权特定计划；应用身份以最小权限执行变更；验证随后查询实际资源并测试预期行为。这些阶段都不应在无提示的情况下合并成“运行流水线”。

计划是有时效的证据，不是承诺。生成计划后，另一个执行者可能更改资源，提供商容量可能变化，策略可能分配，机密可能过期。对于高风险变更，应应用已保存的计划，或在执行前立即重新规划并拒绝实质差异。证据包应标识配置提交、提供程序版本、计划摘要、审批人、执行身份、部署 ID、资源 ID 和验证结果。

退役属于同一工作流。预览破坏性操作，对保留数据和共享依赖项分类，获得所需审批，执行删除或分离语义，并验证可计费或暴露资源已不存在。如果公网 IP、磁盘、快照、DNS 记录、密钥、部署身份或保留的存储对象仍然存在，“销毁命令返回成功”并不够。只有清理路径经过测试，临时基础设施才真正是临时的。

阶段	最低控制	证据
格式化与验证	语法、架构、模块输入、提供程序兼容性	验证输出
计划 / What-If	明确作用域，以及预测的创建/更新/替换/删除操作	已保存计划或渲染后的变更报告
策略与安全	允许区域、标签、暴露、身份、加密、高风险替换	策略结果与例外
成本审查	估算重要的周期性、使用、出口流量与承诺影响	成本增量或有文档说明的限制
审批	针对特定计划摘要与环境指定审批人	审批记录
应用	最小权限的非人员身份与串行化状态访问	部署/运行 ID 与活动证据
验证	查询实际资源图，并运行功能/安全检查	清单与测试输出
退役	销毁/取消管理预览、数据保留决策、残留资源检查	清理与零残留证据

4.11 跨云转译组织与 IaC

多云比较应保留工程意图，而不是强行统一层次结构。Azure 资源组、阿里云资源组、腾讯云项目和火山引擎项目都支持分组，但它们与账户、策略、成本和服务覆盖范围的关系不同。标签继承、资源清单和原生编排也应同样谨慎处理。正确的转译既记录最近似实现，也记录影响运维的非等价性。

Terraform 可以在这些提供商之间提供通用存储库与计划/状态工作流，但每个提供商仍需要自己的资源类型、身份验证、区域标识符、生命周期行为和功能覆盖。原生编排仍然重要：Azure Resource Manager 与 Bicep、阿里云 ROS、火山引擎基础设施管理公开提供商专用控制平面能力；腾讯云通常使用 Terraform 提供程序与云服务 API。架构应为每项资源选择一个主要期望状态来源，而不能让原生模板、Terraform 和门户编辑在没有所有权规则时相互竞争。

工程意图	Azure	阿里云	腾讯云	火山引擎
多账户层次结构	管理组与订阅	资源目录与成员账户	腾讯云企业组织部门与成员账户	云管理 / 多账户资源服务
工作负载分组	资源组	资源组	项目与产品专用分组结构	项目
横向元数据	通过治理工具管理资源标签与标签策略	标签与标签策略，包括受支持的继承/修正	标签、标签策略与成本分摊标签	标签与项目元数据
清单	Azure Resource Graph	资源中心 / Cloud Config / 标签搜索	资源中心 / Config / 标签搜索	资源中心 / 标签搜索
原生 IaC	ARM 模板、Bicep、部署堆栈	ROS 模板与 ROS 中的托管 Terraform	服务 API 与提供商工具；Terraform 是常见路径	基础设施管理和受支持的服务专用模板
可移植 IaC	HashiCorp AzureRM provider	Alibaba Cloud provider	Tencent Cloud provider	Volcengine provider
成本归属	订阅/资源组作用域、标签、Cost Management 继承/分摊	账户/资源组/标签与成本中心	项目、分摊标签、计费与 FinOps 工具	项目、标签、计费与成本管理服务
要记录的非等价性	资源组生命周期与部署堆栈行为	资源目录、账户、资源组与标签策略模式之间的区别	项目覆盖与服务专用资源分组	项目覆盖、标签支持与基础设施管理/提供程序成熟度

多云规则

先标准化元数据词汇与变更证据，再标准化模板。共享的所有者、工作负载、环境、成本中心和生命周期契约，往往比针对每一种提供商资源类型的强制抽象更具可移植性。

4.12 实验 1：资源组织与成本归属

本实验创建专用 Azure 资源组和低成本存储账户，对二者应用相同的受控元数据，通过 Azure Resource Graph 查询资源，准备成本归属视图并证明清理。它与本书其他实验相互独立。目标不是存储数据，而是证明可以从控制平面验证作用域、所有权、清单和成本上下文。

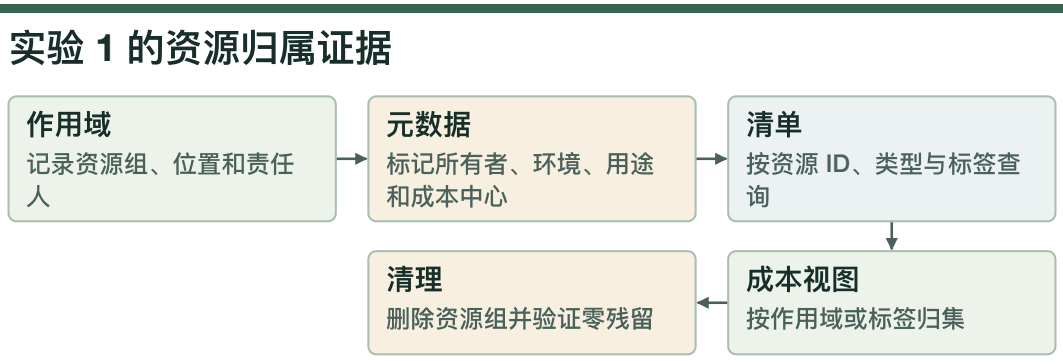


图 4-5 只有当组织与归属由可重复证据包和清理证明表示时，实验 1 才通过验收。

项目	要求
权限	在一个 Azure 订阅中创建和删除资源组及存储账户的权限；对所选作用域中 Resource Graph 与 Cost Management 的读取权限
工具	Azure CLI；当前 CLI 版本首次使用 az graph 时会自动安装 Resource Graph 扩展
成本边界	一个不上传数据的 Standard_LRS StorageV2 账户。证据包完成后删除资源组
元数据契约	owner, workload, environment, purpose, costCenter, lifecycle, managedBy
证据目录	创建本地 evidence/ 目录。不要在其中放入凭据、令牌或连接字符串
时间注意事项	成本与使用数据不是实时的。基于标签的成本结果只有在使用量产生并处理后才会出现；标签不追溯既往记录

```
# 1. Choose a subscription and lab variables
az login
az account set --subscription "<subscription-id-or-name>"

LOCATION="eastasia"          # choose an allowed region
```

```
SUFFIX="$(openssl rand -hex 3)"
RG="csaip-ch04-${SUFFIX}-rg"
STORAGE="csaip${SUFFIX}sa"
OWNER="team-cloud-course"

mkdir -p evidence

# 2. Create a dedicated scope with controlled metadata
az group create   --name "$RG"   --location "$LOCATION"   --tags owner="$OWNER"
workload="book-ch04"          environment="lab"          purpose="resource-
organization"          costCenter="training"          lifecycle="ephemeral"
managedBy="cli"   --output json > evidence/01-resource-group.json

# 3. Create one low-cost resource with the same metadata
az storage account create   --name "$STORAGE"   --resource-group "$RG"   --
location "$LOCATION"   --sku Standard_LRS   --kind StorageV2   --min-tls-version
TLS1_2   --allow-blob-public-access false   --tags owner="$OWNER"
workload="book-ch04"          environment="lab"          purpose="resource-
organization"          costCenter="training"          lifecycle="ephemeral"
managedBy="cli"   --output json > evidence/02-storage-account.json
```

```
# 4. Query the resource group itself
az graph query -q "
ResourceContainers
| where type =~ 'microsoft.resources/subscriptions/resourcegroups'
| where name =~ '$RG'
| project id, name, type, location, tags
" -o json > evidence/03-scope-query.json

# 5. Query all resources and normalize the required tags
az graph query -q "
Resources
| where resourceGroup =~ '$RG'
| project id, name, type, location,
    owner=tostring(tags.owner),
    workload=tostring(tags.workload),
    environment=tostring(tags.environment),
    purpose=tostring(tags.purpose),
    costCenter=tostring(tags.costCenter),
    lifecycle=tostring(tags.lifecycle),
    managedBy=tostring(tags.managedBy)
| order by type asc
" -o json > evidence/04-inventory.json

# 6. The following query must return an empty result set
az graph query -q "
Resources
```

```
| where resourceGroup =~ '$RG'
| where isempty(tostring(tags.owner))
    or isempty(tostring(tags.workload))
    or isempty(tostring(tags.environment))
    or isempty(tostring(tags.costCenter))
    or isempty(tostring(tags.lifecycle))
| project id, name, type, tags
" -o json > evidence/05-tag-gaps.json
```

在资源组作用域打开 Cost Management。保存一个筛选到实验资源组的 Cost Analysis 视图。使用数据出现后，按 costCenter 或 workload 标签分组，并导出表格或捕获已保存视图。如果尚未产生费用，应记录时间、作用域、筛选器和预期标签维度，而不要声称已实现实时归属。Cost Management 数据会随着提供程序发出使用量而刷新，新应用的标签不会改写历史记录。

审查证据文件。权威验收条件并不是门户显示绿色勾选，而是作用域查询返回资源组、清单返回存储账户与必需元数据、标签缺口查询不返回任何资源，并且成本视图使用同一分摊契约设定作用域或分组。

```
# 7. Cleanup is part of the lab
az group delete --name "$RG" --yes
az group wait --name "$RG" --deleted

# 8. Prove that the scope no longer exists
az group exists --name "$RG" | tee evidence/06-group-exists-after-delete.txt

# Optional: prove that no resource remains visible under the group name
az graph query -q "
Resources
| where resourceGroup =~ '$RG'
| project id, name, type
" -o json > evidence/07-resources-after-delete.json
```

验收证据	通过条件
01-resource-group.json	资源组 ID、区域和必需标签均存在
04-inventory.json	每项实验资源都有提供商 ID、类型、位置以及标准化所有权/成本元数据
05-tag-gaps.json	结果包含零项资源
成本视图或导出	已记录资源组作用域和预期标签分组；已披露延迟
06-group-exists-after-delete.txt	值为 false

验收证据	通过条件
07-resources-after-delete.json	控制平面传播完成后，结果包含零项资源

中国云转译

重复意图而不是重复命令：创建隔离的组/项目，应用相同的受控标签，查询提供商资源中心，打开成本作用域或分摊标签视图，并删除组/项目资源。记录提供商无法以相同方式分组、继承或分摊标签的位置。

4.13 实验 2：IaC 生命周期——创建、验证、变更与销毁

本实验使用带 AzureRM 提供程序的 Terraform，因为状态和计划生命周期清晰可见，而且同一工作流以后可以转译到其他提供商。Bicep 仍是本章的 Azure 原生参考。实验创建资源组与存储账户，保存计划并应用，进行一次受控就地变更，验证实际资源图，预览销毁、销毁堆栈，并证明没有托管资源残留。

实验有意避免使用密码、数据平面密钥和已上传对象。即便如此，Terraform 状态仍属于敏感运维数据，绝不能提交。请使用一次性目录。在团队环境中，用加密且带锁的远程后端替代本地状态。

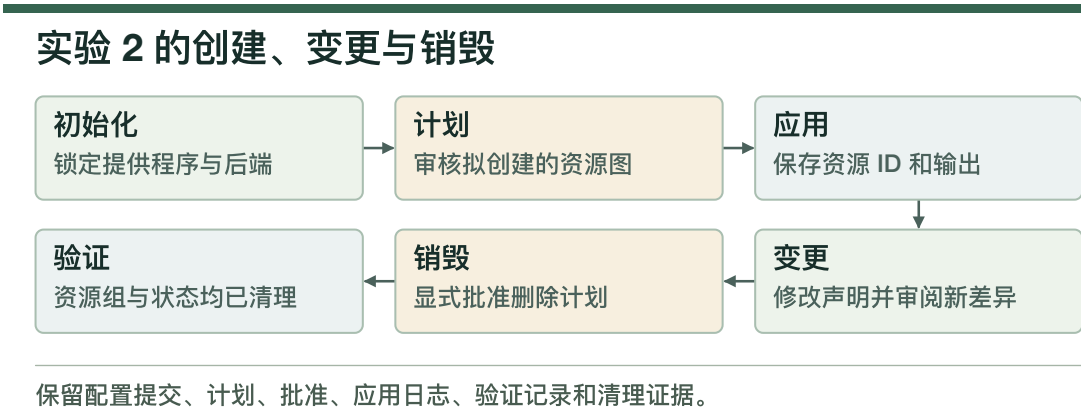


图 4-6 实验 2 把销毁与零残留验证视为验收标准，而非可选清理。

文件	责任
versions.tf	Terraform 与提供程序约束
main.tf	提供程序配置、标签、资源组、存储账户与输出
variables.tf	订阅、区域、后缀、所有者、成本中心和受控访问层级

文件	责任
terraform.tfvars	本地非机密输入；不要提交
.gitignore	排除状态、计划、崩溃日志和真实变量文件
evidence/	存储人类可读的计划、输出、清单查询和清理证明

```
# versions.tf
terraform {
  required_version = ">= 1.8.0"

  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = ">= 5.0, < 6.0"
    }
  }
}

provider "azurerm" {
  features {}
  subscription_id = var.subscription_id
}
```

```
# variables.tf
variable "subscription_id" { type = string }
variable "location" {
  type    = string
  default = "eastasia"
}
variable "suffix" {
  type = string
  validation {
    condition     = can(regex("^[a-z0-9]{6}$", var.suffix))
    error_message = "suffix must contain exactly six lowercase letters or
digits."
  }
}
variable "owner"       { type = string }
variable "cost_center" { type = string }
variable "access_tier" {
  type    = string
  default = "Hot"
  validation {
    condition     = contains(["Hot", "Cool"], var.access_tier)
```

```
    error_message = "access_tier must be Hot or Cool."
  }
}
```

```
# main.tf
locals {
  common_tags = {
    owner       = var.owner
    workload    = "book-ch04-iac"
    environment = "lab"
    purpose     = "iac-lifecycle"
    costCenter  = var.cost_center
    lifecycle   = "ephemeral"
    managedBy   = "terraform"
  }
}

resource "azurerm_resource_group" "lab" {
  name     = "csaip-ch04-${var.suffix}-rg"
  location = var.location
  tags     = local.common_tags
}

resource "azurerm_storage_account" "lab" {
  name                            = "csaip${var.suffix}sa"
  resource_group_name             = azurerm_resource_group.lab.name
  location                        = azurerm_resource_group.lab.location
  account_tier                    = "Standard"
  account_replication_type        = "LRS"
  account_kind                    = "StorageV2"
  access_tier                     = var.access_tier
  min_tls_version                 = "TLS1_2"
  allow_nested_items_to_be_public = false
  tags                            = local.common_tags
}

output "resource_group_name" {
  value = azurerm_resource_group.lab.name
}
output "storage_account_id" {
  value = azurerm_storage_account.lab.id
}
```

```
# terraform.tfvars.example
subscription_id = "00000000-0000-0000-0000-000000000000"
location        = "eastasia"
```

```
suffix          = "a1b2c3"
owner           = "team-cloud-course"
cost_center     = "training"
access_tier     = "Hot"

# .gitignore
.terraform/
*.tfstate
*.tfstate.*
*.tfplan
crash.log
terraform.tfvars
evidence/*.json
```

```
# 1. Initialize, validate, and create a saved plan
mkdir -p evidence
cp terraform.tfvars.example terraform.tfvars
# Edit subscription_id and choose a unique six-character suffix.

az login
terraform init
terraform fmt -check -recursive
terraform validate
terraform plan -out=create.tfplan
terraform show -no-color create.tfplan   | tee evidence/01-create-plan.txt

# 2. Apply exactly the reviewed plan
terraform apply create.tfplan   | tee evidence/02-create-apply.txt
terraform output -json > evidence/03-outputs.json
RG="$(terraform output -raw resource_group_name)"

# 3. Verify actual control-plane state
az graph query -q "
Resources
| where resourceGroup =~ '$RG'
| project id, name, type, location, tags
" -o json > evidence/04-inventory-after-create.json
```

在 terraform.tfvars 中把 access_tier 设置为 "Cool"，进行一次受控变更。运行新计划并检查提供程序建议就地更新还是替换。对于此配置，预期结果通常是就地更新存储账户，但已安装提供程序版本生成的计划才是权威。如果计划包含无法解释的替换或删除，请勿继续。

```
# 4. Plan and apply the controlled change
terraform plan -out=change.tfplan
terraform show -no-color change.tfplan   | tee evidence/05-change-plan.txt
terraform apply change.tfplan   | tee evidence/06-change-apply.txt

# 5. Preview destruction before authorizing it
terraform plan -destroy -out=destroy.tfplan
terraform show -no-color destroy.tfplan   | tee evidence/07-destroy-plan.txt

# 6. Apply the reviewed destroy plan
terraform apply destroy.tfplan   | tee evidence/08-destroy-apply.txt

# 7. Verify zero remainder
az group exists --name "$RG"   | tee evidence/09-group-exists-after-destroy.txt
terraform state list   | tee evidence/10-state-list-after-destroy.txt
```

验收检查	必需结果
创建计划	只创建预期资源组与存储账户
创建并应用	输出包含提供商资源 ID，且清单查询找到资源
变更计划	计划解释访问层级变更，且不包含无法解释的替换或删除
销毁计划	每项托管资源都显示为计划删除
销毁并应用	应用完成，且没有隐藏的保留子资源
控制平面验证	az group exists 返回 false
状态验证	terraform state list 不返回任何托管地址

提供程序版本说明

印刷书中的提供程序约束是经过测试的范围，不是永久建议。出版前以及在配套存储库中，应使用当前稳定版 Terraform 与提供程序重新运行实验。当实验纳入版本控制时，提交 terraform init 生成的提供程序锁文件。

多云转译

保留工作流并替换提供商契约：声明最近似的组/项目和一个低成本资源；固定提供程序；计划；应用；查询提供商清单；进行一次受控变更；预览销毁；销毁；证明零残留。记录其提供商生命周期无法安全表达的任何资源。

4.14 CloudDesk 复用：组织、标签审计、部署与差异

CloudDesk FinOps 智能体以后可以复用本章建立的能力，但本章不实现智能体。只读能力比较直接：列出某个作用域的清单、规范化所有权元数据、计算标签覆盖率，并解释漂移或建议计划。变更能力需要更强控制，因为创建作用域或应用 IaC 会产生成本、改变暴露并可能删除数据。

安全的智能体工具绝不能接收通用 Terraform shell。工具代理应公开有边界的能力，使用获批模块、类型化输入、固定作用域、默认仅计划、计划摘要、风险分类，并分别审批应用或销毁。结果必须包含提供商资源 ID 和证据链接，而不是无结构的成功消息。

```
capability: cloud.deploy.stack
risk_level: R3_mutating_reversible_with_review
inputs:
  provider: azure
  module_id: approved/private-storage-v2
  target_scope: /subscriptions/.../resourceGroups/...
  parameters:
    owner: team-platform
    workload: finops-agent
    environment: dev
    costCenter: cc-4102
mode_default: plan_only
controls:
  - validate target scope and required tags
  - pin module and provider versions
  - return plan digest and destructive-operation summary
  - require approval bound to the plan digest before apply
  - use a least-privilege deployment identity
  - verify resource IDs, tags, policy state, and cost scope
  - require a separate destroy approval
outputs:
  plan_digest: sha256:...
  resource_ids: []
  evidence_links: []
  residual_resources: []
```

能力	用途	风险与最低控制
cloud.resource.organize	在获批作用域中创建或移动资源，并应用必需元数据	R2/R3；获批目标作用域、策略验证、预览、回滚限制
cloud.cost.tagging.audit	衡量缺失、无效、陈旧以及未进行成本分摊的标签	R0 只读；Resource Graph/资源中心与计费读取权限
cloud.deploy.stack	规划并应用获批的 Bicep/Terraform 模块	R3；默认仅计划、基于摘要审批、最小权限执行、验证

能力	用途	风险与最低控制
cloud.resource.diff	比较声明状态、工具已知状态与提供商实际状态	R0/R1；只读清单与状态元数据，不自动修正
cloud.resource.retire	预览并执行有边界的退役，同时作出保留决策	破坏性数据路径为 R4；独立审批、备份/保留检查、零残留证明

4.15 关键要点

- 资源组织是一项控制平面设计决策，会塑造权限、策略继承、计费、影响范围、证据和生命周期。
- 组织层次、账户/订阅、资源组/项目、资源 ID、名称和标签分别解决不同问题，不能被压缩成一种命名约定。
- 标签是受治理的元数据模型。必需键需要受控值、记录系统、消费者、合规度量和定期对账。
- 成本分摊结合层次结构、标签、提供商计费数据、受支持的继承，以及针对共享或未分摊费用的明确规则。
- 可查询清单连接资源 ID、所有权、类型、位置、生命周期和管理元数据；活动日志、遥测与计费回答不同问题。
- IaC 不只是配置文本，而是期望状态、提供程序架构、执行身份、状态或部署元数据、依赖关系图、证据、验证与退役之间的契约。
- 原生 IaC 与 Terraform 提供不同的状态和生命周期模型。Terraform 标准化的是工作流，而不是跨云资源语义。
- 状态、计划制品、变量文件和部署历史可能包含敏感运维数据，需要显式保护。
- 必须先分类漂移再对账。有些差异应还原，有些应导入，有些应委派给另一个控制器，还有些应记录为紧急变更。
- 销毁与零残留验证是临时基础设施的验收标准，而不是可选的日常清理。
- 两个实验彼此独立：一个证明组织与成本归属，另一个证明由声明状态驱动的创建—变更—销毁生命周期。
- CloudDesk 可以复用有边界的组织、标签审计、部署、差异和退役能力，但通用 shell 或不受限制的 Terraform 访问不是安全的智能体工具。

参考资料与官方文档

[1] 美国国家标准与技术研究院：《NIST 云计算定义》。NIST SP 800-145，2011。https://doi.org/10.6028/NIST.SP.800-145

[2] Microsoft：资源组织设计领域概述。《云采用框架》。更新于 2026 年 1 月 9 日。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/design-area/resource-org

[3] Microsoft：有效组织 Azure 资源。《云采用框架》。更新于 2025 年 10 月 3 日。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-setup-guide/organize-resources

[4] Microsoft：定义命名约定。《云采用框架》。更新于 2026 年 6 月 25 日。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming

[5] Microsoft：定义标签策略。《云采用框架》。更新于 2025 年 7 月 1 日。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-tagging

[6] Microsoft：使用标签组织 Azure 资源与管理层次结构。Microsoft Learn。更新于 2025 年 9 月 15 日。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/tag-resources

[7] Microsoft：Azure 资源的标签支持。Microsoft Learn。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/tag-support

[8] Microsoft：Azure Resource Graph 概述。Microsoft Learn。https://learn.microsoft.com/en-us/azure/governance/resource-graph/overview

[9] Microsoft：Azure Resource Manager 的 Azure Resource Graph 示例查询。更新于 2026 年 2 月 27 日。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/resource-graph-samples

[10] Microsoft：快速入门——使用 Azure CLI 运行 Resource Graph 查询。https://learn.microsoft.com/en-us/azure/governance/resource-graph/first-query-azurecli

[11] Microsoft：成本分摊简介。Microsoft Cost Management。更新于 2025 年 6 月 27 日。https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/cost-allocation-introduction

[12] Microsoft：使用标签继承进行成本分组与分摊。更新于 2025 年 6 月 27 日。https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/enable-tag-inheritance

[13] Microsoft：Cost Management 中的常见成本分析用途。更新于 2025 年 7 月 1 日。https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/cost-analysis-common-uses

[14] Microsoft：了解 Cost Management 数据。Microsoft Learn。
https://learn.microsoft.com/en-us/azure/cost-management-billing/costs/understand-cost-mgt-data

[15] Microsoft：Azure Resource Manager 模板概述。更新于 2026 年 6 月 26 日。
https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview

[16] Microsoft：Bicep 文件结构与语法。Microsoft Learn。
https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/file

[17] Microsoft：使用 Azure CLI 部署 Bicep 文件。更新于 2026 年 5 月 27 日。
https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deploy-cli

[18] Microsoft：Bicep What-If——部署前预览变更。更新于 2026 年 3 月 3 日。
https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deploy-what-if

[19] Microsoft：使用 Bicep 创建和部署 Azure 部署堆栈。更新于 2026 年 6 月 24 日。
https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/deployment-stacks

[20] Microsoft：Azure Resource Manager 部署模式。更新于 2026 年 6 月 26 日。
https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deployment-modes

[21] HashiCorp：Terraform 状态。
https://developer.hashicorp.com/terraform/language/state

[22] HashiCorp：Terraform 状态的用途。
https://developer.hashicorp.com/terraform/language/state/purpose

[23] HashiCorp：terraform plan 命令参考。
https://developer.hashicorp.com/terraform/cli/commands/plan

[24] HashiCorp：terraform apply 命令参考。
https://developer.hashicorp.com/terraform/cli/commands/apply

[25] HashiCorp：terraform destroy 命令参考。
https://developer.hashicorp.com/terraform/cli/commands/destroy

[26] HashiCorp：管理资源漂移。
https://developer.hashicorp.com/terraform/tutorials/state/resource-drift

[27] HashiCorp：资源生命周期元参数。
https://developer.hashicorp.com/terraform/language/meta-arguments/lifecycle

[28] HashiCorp：在配置中使用模块。
https://developer.hashicorp.com/terraform/language/modules/configuration

[29] HashiCorp：重构模块。
https://developer.hashicorp.com/terraform/language/modules/develop/refactoring

[30] FinOps Foundation：分摊能力。
https://www.finops.org/framework/capabilities/allocation/

[31] FinOps Foundation：《云成本分摊指南》。https://www.finops.org/wg/cloud-cost-allocation/

[32] FinOps Foundation：如何衡量标签策略合规性。更新于 2026 年 3 月 17 日。
https://www.finops.org/wg/how-to-measure-tagging-policy-compliance/

[33] FOCUS：FinOps 开放成本与使用规范 1.4。
https://focus.finops.org/docs/specification/v1-4/

[34] 阿里云：资源组织与管理模式比较。
https://www.alibabacloud.com/help/en/resource-management/product-overview/differences-and-relationships-among-the-resource-directory-resource-group-and-tag-services

[35] 阿里云：标签策略概述。更新于 2026 年 6 月 3 日。
https://www.alibabacloud.com/help/en/resource-management/tag/user-guide/overview

[36] 阿里云：启用资源组自动标签继承。更新于 2026 年 6 月 3 日。
https://www.alibabacloud.com/help/en/resource-management/tag/user-guide/enable-automatic-tag-inheritance-from-a-resource-group

[37] 阿里云：资源编排服务文档。https://www.alibabacloud.com/help/en/ros/

[38] 阿里云：在 ROS 中创建 Terraform 模板。更新于 2026 年 6 月 2 日。
https://www.alibabacloud.com/help/en/ros/user-guide/create-a-terraform-template

[39] 腾讯云：腾讯云企业组织文档。
https://www.tencentcloud.com/document/product/1031

[40] 腾讯云：项目管理。https://www.tencentcloud.com/document/product/378/34726

[41] 腾讯云：标签管理。
https://www.tencentcloud.com/document/product/1172/52380

[42] 腾讯云：标签策略概述。
https://www.tencentcloud.com/document/product/1031/55959

[43] 腾讯云：成本分摊标签。更新于 2026 年 5 月 15 日。
https://www.tencentcloud.com/document/product/555/32276

[44] 腾讯云：使用 Terraform 管理现有资源。
https://www.tencentcloud.com/document/product/1172/52393

[45] 火山引擎：项目资源管理。https://www.volcengine.com/docs/6649/94337

[46] 火山引擎：管理标签与资源。更新于 2025 年 2 月 11 日。
https://www.volcengine.com/docs/6649/189381

[47] 火山引擎：资源中心概述。更新于 2025 年 7 月 16 日。
https://www.volcengine.com/docs/6649/1109337

[48] 火山引擎：Terraform 概念与提供程序文档。更新于 2026 年 3 月 17 日。
https://www.volcengine.com/docs/6706/2275245

[49] 火山引擎：基础设施管理产品介绍。更新于 2026 年 8 月 11 日。
https://docs.volcengine.com/docs/88540?lang=zh

[50] 火山引擎：创建基础设施管理资源栈。更新于 2026 年 7 月 9 日。
https://www.volcengine.com/docs/88540/2549851

草稿边界说明

本章建立资源组织与 IaC 生命周期纪律，而不是完整的 Bicep 或 Terraform 语言教程。后续章节会把这套组织模型应用于计算、网络、存储、应用 PaaS、Kubernetes、GitOps、可观测性、FinOps 和 AI PaaS。自定义无服务器智能体运行时与租户隔离实现属于另一本书《无服务器 AI 智能体开发》。

第二篇 / 第 5 章

身份与最小权限

人员、工作负载和未来智能体如何获得作用域严格受限的权限

身份是控制平面的首要边界。程序应获得身份与短期授权，而不是嵌入式凭据；每项权限都应能从任务、操作、资源作用域、条件和期限方面得到解释。

本章内容

- 为何身份是云控制平面的首要安全边界
- 身份验证、授权、凭据、委派和审计证据有何不同
- 为何最小权限是一条权限包络，而不是一个角色名称
- 为何应区分人员身份、工作负载身份、托管身份和联合身份
- Azure RBAC 如何组合主体、角色定义、作用域与可选条件
- 为何控制平面权限不会自动授予数据平面权限
- Key Vault、临时令牌、工作负载身份联合和 Privileged Identity Management 如何减少常驻机密与常驻特权
- 如何在部署、摄取、检索、运维和智能体工具执行之间划分身份
- RAM、CAM 与火山引擎 IAM 如何转译同一工程意图
- 如何在不向工作负载传递客户端机密的情况下，创建并验证只读服务身份

部署流水线因为需要创建一个资源组而被授予订阅作用域的 Owner。Web 应用也使用同一个服务主体，因为凭据已经可用。摄取作业、检索 API、管理仪表板和一个实验智能体都复用该凭据。机密被复制到多个 CI 变量和应用设置中。系统可以工作，却不存在任何可辩护的权限边界。

如果机密泄露，攻击者获得的不是某项任务所需的权限，而是所有捷径累积起来的权限：资源创建、删除、角色分配、数据访问、机密检索，甚至跨环境访问。如果智能体遭到提示词注入，模型输出就可能变成在同一宽泛身份下发出的请求。问题并非没有进行身份验证，而是身份验证证明了调用者控制某个身份，而该身份的权限从未经过设计。

本章把身份视为云资源系统，而不是登录功能。核心问题不只是“此调用者能否获得令牌？”，而是“哪个主体可以在什么条件下、以多长时间，对哪项资源执行哪种操作？什么证据能够证明该决策？”该问题适用于管理员、开发者、流水线、虚拟机、函数、Kubernetes 工作负载、AI 摄取进程、检索应用以及未来的智能体工具执行器。

本章论点

最小权限不是通过选择一个听起来安全的角色名称实现的；只有当主体、允许操作、资源作用域、条件、凭据路径和期限都不超过已验证任务的需要时，才真正实现。

从主体到可追溯的访问决策



凭据证明身份，授权决策约束操作，审计证据连接调用者与结果。

图 5-1 身份连接身份验证、授权、操作和可问责证据。

5.1 身份是首要控制平面边界

第一章把云描述为可编程控制平面。每项有意义的管理请求都必须来自某个主体。门户让人员登录；CLI 或基础设施流水线获取令牌；托管应用调用 API；虚拟机、函数、容器或 Kubernetes Pod 访问下游资源。控制平面在接受创建、更新、分配、扩缩容或删除等生命周期操作前，会先评估调用者。

网络隔离、加密、策略与监控仍然不可或缺，但身份决定谁可以跨越这些边界并更改其配置。私有终结点让服务不在公网上可达，但受信网络内权限过大的主体仍可删除或重新配置资源；加密保护静态数据，但有权检索明文的主体可以合法解密；策略可以阻止不合规资源状态，而谁有权豁免或修改策略仍然是身份问题。

因此，应在应用连线之前设计身份。依赖关系图应说明哪个工作负载以什么身份访问哪项资源。部署身份应与运行时身份不同；读取路径不应暗中带有写权限；监控身份不应能修改被观测的工作负载；工具执行器不应能为自己授予更多角色。

表 5-1 本书使用的身份词汇。

术语	持久含义	工程问题
主体	请求访问的人员、组、工作负载、服务、角色或托管身份。	是谁或什么对象在执行操作？

术语	持久含义	工程问题
凭据	用于获得或呈现身份验证结果的证明材料：密码、密钥、证书、外部令牌或平台托管凭据。	主体如何证明控制权，同时不暴露可重复使用的机密？
令牌	身份验证后签发、向资源或授权服务出示的有时限断言。	令牌携带哪个受众、声明、期限和委派权限？
身份验证	验证声明者控制其声称的身份。	调用者是否确实是其声称的主体？
授权	决定经过身份验证的主体可以对合格资源执行某项操作。	该主体此时此地能否执行此操作？
角色定义 / 策略	允许和排除操作的具名或结构化集合，通常包括资源与条件语法。	描述了哪些操作？
角色分配 / 绑定	在某个作用域把主体与角色或策略关联起来。	权限适用于何处，如何继承？
委派	向另一个主体分配、代入或传递权限的权限。	该身份能否创建超过任务需要的更多权限？
审计证据	登录、授权、活动、数据访问和角色变更记录。	能否重建决策及其影响？

5.1.1 身份验证不等于授权

身份验证回答身份问题，授权回答权限问题。OAuth 2.0 是面向有限访问的授权框架；OpenID Connect 增加了用于验证最终用户身份的身份层。在云工程中，这些协议可能参与登录或令牌交换，但资源仍会评估最终主体可以做什么。有效令牌可以通过身份验证，同时被正确拒绝访问。

这一区分对运维很有用。401 类失败通常表示身份验证缺失或不可接受；403 类失败通常表示调用者已知但未获授权。确切状态与错误契约因服务而异，但诊断顺序稳定：识别主体，检查令牌受众与声明，确定请求的操作与资源，枚举有效分配与条件，再检查策略或数据平面控制。

5.1.2 凭据不是身份

客户端机密不是服务主体，AccessKey 对不是 RAM 用户，密码也不是人员。凭据是与身份关联、可替换的证明机制。把凭据当作身份会造成共享密钥、机密复制、所有权不清和生命周期缺失。安全设计先赋予主体名称和用途，再为环境选择风险最低的身份验证机制。

最佳凭据往往是应用无法读取或导出的凭据。Azure 托管身份是一种特殊服务主体，其凭据由平台管理；应用无需存储机密即可请求短期 Microsoft Entra 令牌。工作负载身份联合让 GitHub Actions 作业或 Kubernetes 服务账户等外部工作负载，用受信 OIDC 令牌交

换云访问令牌，而无需维护云客户端机密。阿里云 RAM 角色、腾讯云 CAM 角色和火山引擎 IAM 角色也在代入角色后使用临时凭据，但具体工作负载集成和服务覆盖范围不同。[5] [6] [14] [17] [20]

5.2 最小权限是一条权限包络

NIST 把最小权限定义为：把用户或进程限制在履行其功能所需的最低系统授权和资源范围内。这一定义包含云设计经常错误拆开的两个概念：授权必须最小，符合条件的资源范围也必须最小。覆盖整个组织的只读角色，可能比仅覆盖一项资源且作用域狭窄的操作员角色暴露更多信息；审批后只激活五分钟的强大角色，可能比永久分配给每位工程师的中等角色造成更少常驻风险。

一种有用的云表达是：

```
有效权限 = 主体 × 操作 × 资源作用域 × 条件 × 时间
```

该表达是概念性的，并非算术公式，但乘法直觉很重要。扩大任何维度都会扩大影响范围。只有每个维度都有明确理由和否定测试时，才达到最小权限。



图 5-2 最小权限围绕一项获授权任务，约束主体、操作、作用域、条件和时间。

5.2.1 主体：一种用途，一个生命周期

为不同的运维目的使用不同主体。部署流水线、应用运行时、摄取工作进程、检索 API、监控收集器和紧急管理员具有不同生命周期与证据要求。复用一个身份会掩盖这些差异。审计事件中出现某个角色时，运维人员应能从身份名称与所有权元数据推断其用途。

直接向个人分配权限可用于特殊任务，但团队权限通常应通过组或受治理角色管理。组成员关系为入职、调动和离职提供生命周期边界。即使平台不强制这些字段，工作负载身份也应具有所有者、环境、应用以及到期或复核元数据。

5.2.2 操作：分离工作职责与数据访问

角色定义应表达某项工作职责所需的操作。不要从 Owner、Administrator 或通配符开始，再减去几个显而易见的危险操作。应从任务开始，枚举提供商操作。适用时优先使用内置工作职责角色；只有当内置选项实质上过宽时才创建自定义角色，并保留为每项操作提供依据的分析。

必须分别关注管理操作与数据操作。创建存储账户是管理操作，读取 Blob 是数据操作；创建 Key Vault 是管理操作，读取机密值是数据操作；创建模型部署是管理操作，发送推理请求是数据操作。只检查管理操作的权限审查，可能批准无法运行的工作负载，也可能遗漏能够读取敏感数据的身份。

5.2.3 作用域：继承既方便又危险

云权限通常沿组织或资源层次向下继承。管理组、组织、账户、订阅、项目或资源组上的角色可以适用于许多后代。继承简化管理，也会把局部合理的角色变成宽泛常驻访问。作用域应是包含任务所需资源的最窄稳定边界。

资源级分配不一定是最佳运维答案。数千项一次性分配可能无法审查。设计目标是既狭窄又可治理：当专用资源组、项目、工作区、保管库或账户边界能够表达内聚应用或环境时，应优先采用。边界应支持委派、成本归属、策略和删除，而不只是 RBAC。

5.2.4 条件与属性：筛选权限，而不是替代权限

基于属性的条件可以通过评估资源、请求、环境或主体属性来收窄角色分配。Azure ABAC 使用角色分配条件扩展了部分 Azure RBAC 场景。其他提供商也在策略语言中公开条件语法，例如限制只能访问带项目标签的对象、把角色限制到源网络，或约束管理员可以分配哪些角色。

条件不应弥补不必要地宽泛的主体或角色。它增加了另一项必须测试、版本管理和审计的授权依赖。安全审查应说明属性缺失、变更或格式错误时会发生什么，并包含否定测试，证明不合格资源会被拒绝。

5.2.5 时间：移除常驻特权

人员管理通常只需要短时间提升权限，而不是永久分配。Microsoft Entra Privileged Identity Management 可以让用户具有 Azure 或目录角色的资格，要求审批、多重身份验证与理由，并在时间窗口结束后自动移除已激活角色。还可以通过提供商角色代入、临时会话和审批工作流实现类似的即时访问设计。[8]

对工作负载而言，时间通过令牌期限与身份生命周期表达。临时令牌降低被捕获凭据的价值，但无法弥补角色权限过大。自动刷新的令牌可以形成连续有效权限。既要审查会话时长，也要审查允许创建会话的常驻角色分配。

最小权限不是“处处只读”

正确问题不是身份能否写入，而是能否准确执行已验证任务。部署身份必须写入选定资源类型；检索身份可能需要数据平面读取而不需要管理写入；成本收集器可以读取使用记录但不能读取业务数据。最小权限是与任务形状相匹配的权限。

5.3 人员、工作负载、服务与智能体身份

表 5-2 不同身份类别具有不同的身份验证与生命周期需求。

身份类别	典型身份验证	主要控制	常见失败
员工身份	无密码或多重身份验证登录、联合、条件访问。	组、工作职责角色、PIM、访问复核、职责分离。	永久管理员角色，以及由自动化复用的个人账户。
应用或服务主体	证书、联合令牌，或无法避免时使用机密。	具名所有者、狭窄角色、凭据轮换、登录监控。	客户端机密被复制到存储库、流水线和应用设置中。
托管身份	平台托管凭据与短期令牌获取。	角色分配、资源附加、生命周期所有权、活动与登录日志。	一个共享身份附加到无关工作负载，或在工作负载退役后残留。
联合工作负载身份	外部 OIDC/SAML 断言交换云令牌。	精确的签发者、主题、受众、存储库/服务账户绑定、短会话。	宽泛信任模式让非预期工作流或命名空间可以代入身份。
云服务角色	提供商服务代入服务关联角色或受信角色。	信任策略、权限策略、服务所有权、变更审计。	把服务角色授予人员，或把其策略复制给无关身份。
未来智能体运行时	工作负载身份加受约束的工具执行器身份。	工具允许列表、审批、请求绑定、任务作用域、证据、禁止自我委派。	向语言模型提供宽泛云凭据或角色分配权限。

5.3.1 人员身份：减少永久管理权限

人员运维需要具备强身份验证的目录身份，但身份验证强度不会减少授权范围。为重复工作职责使用组，把生产管理与日常开发分开，并在平台与许可允许时使特权角色采用资格制或限时制。紧急或破窗身份应数量极少、独立监控、排除在日常工作流之外，并按照有文档的事件规程测试。

职责分离在权限创建环节最重要。编写基础设施模板的人不应自动成为唯一能够审批并部署到生产的人。能够分配角色、修改信任策略或检索签名密钥的身份，应比普通资源参与者具有更高的审查门槛。角色分配管理员往往比工作负载管理员制造更大风险，因为委派会改变以后谁可以操作。

5.3.2 工作负载身份：消除嵌入式凭据

工作负载应以自身身份进行身份验证，而不是冒充开发者。身份应按有意设计的生命周期存在或消失。系统分配托管身份与一个 Azure 资源耦合，并随资源消失；用户分配托管身份是独立 Azure 资源，可以预先授权并附加到多个计算资源。Microsoft 目前建议需要独立生命周期、复用或预授权的 Microsoft 服务场景使用用户分配托管身份。[5]

对 Azure 外部工作负载，工作负载身份联合会对外部签发者建立信任。云身份信任特定签发者、主题和受众，再用外部令牌交换 Microsoft Entra 访问令牌。该设计移除了云机密，但增加了信任策略义务。repo:organization/* 之类的存储库模式比精确环境或分支主题更易配置，也可能危险地过宽。联合是不需要凭据，并非不需要策略。

5.3.3 服务身份：信任与权限彼此独立

云服务可能需要代表客户资源执行操作：托管数据库写入备份、索引器读取数据源、函数读取机密，或 AI 服务调用嵌入部署。信任策略回答谁可以代入角色或使用身份，权限策略回答代入后的身份可以做什么。混淆两者会让角色可由过多主体代入，或执行过多操作。

服务关联角色与提供商托管角色减少设置工作，但仍应出现在清单和变更审查中。其策略可能随服务演进。不要仅仅因为权限看似有用，就把服务关联策略附加到无关人员或工作负载身份。

5.3.4 智能体身份：模型不是主体

智能体系统可以用自然语言规划操作，但不能把语言模型视为安全主体。主体是获取令牌并调用工具的运行时组件。运行时必须在执行前验证请求的操作、资源、参数、审批状态与身份。对这一决策而言，模型输出是不受信任的输入。

本书止于托管云能力边界。配套卷实现 CloudDesk 租户系统、响应绑定、跨租户隔离、桌面审批和自定义工具运行时。持久原则在本章说明：智能体应获得受治理的工具契约，而不是通用云凭据。

5.4 以 Azure 作为参考身份模型

Azure 是有用的教学模型，因为目录身份、Azure 资源角色、托管身份、策略、机密管理、活动证据和 AI PaaS 资源共同形成一套连贯的控制平面词汇。对象模型并不简单，但其中的区分很有价值。



图 5-3 Microsoft Entra 主体在明确作用域上绑定到 Azure 资源角色。

5.4.1 应用对象、服务主体与托管身份

应用注册会创建应用对象，用于描述应用在其主 Microsoft Entra 租户中的定义。服务主体则是该应用在某个租户中的本地身份，是可以应用角色、许可与策略的对象。因此，一个多租户应用可以只有一个应用对象，但在每个接受它的租户中都有一个服务主体。[4]

托管身份是特殊类型的服务主体，由 Azure 资源管理或创建为独立的用户分配资源。它不是应用用户账户，凭据也不会导出给开发者。工作负载使用 Azure 身份库、Microsoft Authentication Library 或集成连接器，从平台环境请求令牌。[5]

这些对象回答不同的生命周期问题。应用对象代表软件定义；服务主体代表租户本地安全身份；托管身份代表绑定到 Azure 资源运行、无需凭据的工作负载身份。把它们都称为“应用”会使授权审查含糊不清。

5.4.2 Azure 角色分配 = 主体 + 角色 + 作用域（+ 条件）

Azure RBAC 通过角色分配授予访问。分配把用户、组、服务主体或托管身份等安全主体，与某个作用域上的角色定义绑定。作用域可以是管理组、订阅、资源组或单个资源，父作用域分配的权限由子作用域继承。可选条件还可以进一步收窄某些分配。[2] [3] [7]

表 5-3 Azure 角色分配的组成部分。

组成部分	问题	审查证据
安全主体	哪个对象会获得权限？	对象 ID、主体类型、所有者、用途、生命周期。
角色定义	描述了哪些 Actions、DataActions 及排除项？	内置或自定义角色 ID、版本化 JSON、提供程序操作。
作用域	哪些资源会继承该分配？	完整 Azure 资源 ID、父层次结构、子资源清单。
条件	哪些属性或委派角色操作会收窄分配？	条件表达式、版本、测试用例、适用操作。
分配状态	访问是活动、合格、限时、继承还是永久状态？	PIM 记录、开始/结束时间、分配来源、审查人。
证据	什么能够证明使用、拒绝、变更与移除？	登录日志、Activity Log、数据日志、相关性 ID、访问复核。

5.4.3 控制平面角色与数据平面角色

Azure 角色定义把管理 Actions 和 NotActions 与数据平面 DataActions 和 NotDataActions 分开，防止管理通配符在无提示的情况下变成读取未来所有数据类型的权限。Reader 分配允许主体检查其作用域中的 Azure 资源配置，但不会自动让主体列出 Blob、读取 Key Vault 机密值、查询数据库、检索搜索文档或调用每个模型终结点。[3] [10]

表 5-4 管理权限与数据权限彼此独立。

资源	管理平面任务	数据平面任务	最小权限含义
存储	读取账户属性、网络、冗余和生命周期设置。	列出、读取、写入或删除 Blob。	Reader 不等于 Storage Blob Data Reader；两种角色都需审查。
Key Vault	创建或配置保管库、网络、诊断和角色分配。	读取机密值、使用密钥签名或检索证书。	Key Vault Contributor 管理保管库，但不能读取机密值。
Azure AI Search	创建服务，配置网络与诊断。	为文档建立索引，或发出搜索/检索查询。	分离部署、摄取与检索身份。
Foundry/模型资源	创建账户/项目或模型部署，并配置配额。	调用推理、评估，或使用智能体/知识数据平面 API。	资源参与者不应同时成为生产推理调用者。
Kubernetes 服务	创建托管集群并附加身份/网络。	创建 Kubernetes 对象或访问应用数据。	Azure RBAC 与 Kubernetes 授权是组合控制平面。

Key Vault 是特别清晰的例子。控制平面访问管理保管库资源，数据平面访问管理密钥、机密与证书。Microsoft 建议使用 Azure RBAC 进行 Key Vault 授权，并提供 Key Vault Secrets User 等数据角色。Key Vault Contributor 可以管理保管库控制平面，但不能读取机密内容。旧式访问策略配置需要特别谨慎，因为权限足够高的控制平面用户可能能够为自己授予数据访问。[10]

5.4.4 托管身份与工作负载身份联合

对于托管在受支持 Azure 资源上的工作负载，托管身份是默认选择。源资源获得身份，目标资源为该身份获得 RBAC 分配，工作负载在运行时请求短期令牌。应用配置中不放置客户端机密。系统分配身份适合单个资源；用户分配身份支持预授权、复用和独立生命周期。[5]

工作负载身份联合是外部构建系统、Kubernetes 工作负载和跨云计算的默认方向。外部身份提供程序签发签名令牌；Microsoft Entra 检查已配置的签发者、主题和受众，再交换访问令牌。它消除了机密存储与轮换，但仍需要精确的信任、令牌受众、角色作用域和撤销设计。[6]



图 5-4 优先使用平台托管身份或令牌联合；存储的凭据是运行风险逐渐升高的后备方案。

5.4.5 人员特权访问与 PIM

订阅作用域的永久 Owner 或 User Access Administrator 分配，会对资源及未来角色分配形成常驻特权。Azure RBAC 指南建议移除不必要的特权分配、优先使用工作职责角色并收窄特权作用域。Microsoft Entra Privileged Identity Management 增加了合格和限时分

配、审批、MFA、理由、通知、访问复核和审计历史。[8] [9]

PIM 不能替代角色设计。激活三十分钟的 Owner，仍比激活狭窄定义的生产部署者角色更宽泛。用 PIM 缩短常驻时间，用 RBAC 减少操作与作用域。角色分配、策略豁免、密钥恢复和生产破窗等高风险操作，应具有独立的激活与证据路径。

5.4.6 RBAC 与 Policy 回答不同问题

Azure RBAC 评估主体能否在作用域执行操作；Azure Policy 评估由此产生的资源状态是否符合分配的规则。工程师可能获准创建存储账户，但 Policy 会因启用了公网访问或缺少必需标签而拒绝请求。反过来，Policy 不会授予工程师发出请求的权限。两套系统相互组合：RBAC 控制执行者与操作，Policy 控制可接受状态。[11]

这一区分可以避免一种常见反模式：授予宽泛角色，再期待策略使其达到最小权限。Policy 可以阻止部分不合规结果，但主体仍可能读取敏感配置、执行合规却具破坏性的操作，或操作策略覆盖范围之外的资源。最小权限始于身份与授权，再叠加防护规则。

5.5 机密不是身份策略

有些系统仍需要密码、API 密钥、数据库凭据、签名材料或第三方令牌。机密管理器可以降低存储和分发这些值的风险，但本身不会创建最小权限身份系统。应用需要一个身份来检索机密，这个最初的身份验证步骤就是“零号机密”问题。

尽可能使用托管身份或联合访问机密存储。只允许工作负载读取所需机密或保管库边界。分离机密管理与机密使用。记录检索、轮换和版本变更。防止机密内容进入日志、追踪、提示词、构建输出或错误消息。按计划轮换，并在疑似泄露后立即轮换。

表 5-5 机密管理责任。

责任	控制平面决策	证据
存储	哪个保管库或机密管理器在哪个区域和环境存储该值?	资源 ID、加密配置、私有终结点、标签。
检索身份	哪个工作负载可以获得该值?	托管/联合主体与数据平面角色分配。
管理	谁可以创建、轮换、恢复或删除机密?	独立管理员角色、PIM 激活、变更事件。
版本与轮换	该值何时变化，依赖方如何采用新值?	版本 ID、轮换作业、分阶段发布、回滚记录。
暴露预防	明文绝不能出现在哪里?	存储库扫描、流水线遮盖、日志脱敏、提示词筛选。

责任	控制平面决策	证据
退役	什么能够证明机密及所有引用均已消失?	已禁用版本、依赖项清单、删除/恢复证据。

机密仍有必要——但不应成为默认云凭据

机密管理器是机密的正确归宿，却不是创建机密的理由。云身份验证优先使用托管身份或工作负载联合；只有对尚无法消除的凭据才使用机密存储。

5.6 把最小权限设计为可重复流程

凭直觉分配权限时，最小权限会失败。无论对象是人员管理员、部署流水线、函数、Kubernetes 工作负载、RAG 摄取作业还是智能体工具执行器，都应使用同一种设计方法。

5.6.1 从任务契约开始

用可观测术语描述任务：“列出资源组 X 中的资源元数据并生成成本归属报告”，而不是“需要 Azure 访问”。识别输入、输出、失败行为、环境、预期频率和任务是否更改状态。如果描述任务时离不开“所有内容”，边界就还不适合授权。

5.6.2 枚举正向与负向操作

列出必须成功的管理与数据操作，再列出相邻但必须失败的操作。检索应用可以查询一个知识库并调用一个模型部署；它应无法重建索引、修改模型部署、分配角色或枚举无关存储。负向要求会成为可执行的授权测试。

5.6.3 选择主体与凭据路径

选择生命周期与任务匹配的身份。员工职能使用组，Azure 托管计算使用托管身份，外部流水线或 Kubernetes 服务账户使用联合身份。不要跨环境共享主体。指明所有者和退役条件。决定如何获取令牌，而不在代码或配置中放入可复用材料。

5.6.4 选择最窄且可治理的作用域

最窄资源不一定是最易维护的边界。当应用资源组、项目、工作区、保管库或专用资源能提供一致的所有权和清理能力时，应优先采用。如果多项资源必须共享一个工作负载身份，应记录分组，并确保无关的未来资源不能悄然加入继承作用域。

5.6.5 优先使用内置角色，再度量缺口

内置角色易于识别，也由提供商维护，但可能比任务更宽。把其操作与任务契约比较。若缺口不可接受，应根据观察到的提供商操作定义自定义角色，而不是猜测通配符。把角色定义作为代码进行版本管理。扩展带通配符的自定义角色前，审查提供商新增的操作。

5.6.6 添加时间、条件与委派边界

人员特权应尽可能采用资格制或限时制。工作负载会话应使用临时令牌。属性条件可以限制对象标签、存储库、容器或角色委派。把 roleAssignments/write、AssumeRole、信任策略更新和机密管理权限视为委派能力。这些操作应使用独立身份与审批路径。

5.6.7 同时证明允许与拒绝路径

预期操作成功并不能证明权限设计已经验证。只有当必需操作成功、禁止操作失败、失败可以归因于授权而不是无关错误，并且证据能识别主体、作用域、操作和相关性 ID 时，才算完成验证。角色变更后以及出版或部署冻结时应重复测试，因为服务操作与内置角色会演进。

5.6.8 审查使用并移除权限

角色分配会不断累积，因为创建属于交付，而移除被当作清理。应把审查和退役作为一等生命周期操作。盘点未使用主体、陈旧凭据、宽泛分配、继承特权、无所有者服务身份和未附加到活动资源的身份。先移除角色分配再删除身份，使证据保持可理解，随后验证所有引用均已消失。

```
identity_design:
  principal: id-cost-reader-prod
  purpose: "Read Azure resource metadata for cost attribution"
  required_actions:
    - Microsoft.Resources/subscriptions/resourceGroups/read
    - "resource provider read operations within rg-finops-prod"
  prohibited_actions:
    - Microsoft.Resources/subscriptions/resourceGroups/write
    - Microsoft.Authorization/roleAssignments/write
    - "all business-data DataActions"
  scope: /subscriptions/<sub>/resourceGroups/rg-finops-prod
  credential: user-assigned managed identity
  lifetime: assignment reviewed every 90 days
  evidence:
    - role assignment ID
    - successful inventory query
    - failed write attempt
    - Activity Log correlation ID
```

5.7 AI PaaS 与智能体应用的身份边界

AI 原生应用延伸而不是取代身份控制平面。模型终结点、知识源、向量索引、内容安全服务、评估数据、工具连接、存储、遥测和成本 API 都公开不同操作。把它们压缩成一个“AI 应用身份”，会形成宽泛且不透明的主体。



图 5-5 部署、摄取、检索、运维与工具执行应使用独立权限。

表 5-6 AI 原生应用的最低身份拆分。

身份	允许	明确排除
AI 资源部署者	通过 IaC 创建/更新获批模型、搜索、网络、诊断和策略资源。	生产推理、业务数据读取、机密值读取、不受限制的角色分配。
知识摄取工作进程	读取获批源位置，调用一个嵌入部署，写入一个目标索引并报告摄取遥测。	变更模型部署、查询无关数据源、更新访问策略、回答用户查询。
检索应用	使用获批检索或知识库终结点，调用获批模型部署，并发出引用和追踪上下文。	索引架构变更、检索范围之外的源浏览、机密管理、角色分配。
评估器	读取指定测试集和捕获输出，提交评估作业，写入评估结果。	生产变更、不受限制的提示词日志访问、部署或策略变更。
操作员 / SRE	读取健康、日志、指标、追踪、配额、成本和部署状态；发起获批回滚。	默认读取业务数据、导出模型密钥、任意工具执行。

身份	允许	明确排除
智能体工具执行器	在获批资源作用域中执行一项经过验证的工具操作，并绑定任务与审批。	通用云 shell、权限管理、自我委派、无关租户或项目资源。

5.7.1 模型访问是数据平面权限

创建或配置模型资源的身份，不应自动成为调用生产推理的身份。应把管理角色与数据平面角色分开。在服务支持时，把配额和成本归属到调用身份或应用作用域。直接 API 密钥适合原型，但资源级 AI PaaS 设计应优先采用目录身份验证、私有网络、无需轮换的工作负载身份和可审计角色分配。

5.7.2 检索授权必须具有确定性

检索身份可以查询共享知识资源，但必须由受信授权上下文和筛选策略强制文档资格。语言模型不得自行构造租户、工作区或 ACL 谓词。第十八章会展开知识资源生命周期；身份规则很简单：由经过身份验证的应用上下文创建授权筛选器，并通过负向测试证明不合格文本块绝不会进入模型可见上下文。

5.7.3 工具权限必须窄于智能体意图

智能体可以请求“优化云成本”，但该意图不是权限。工具契约应公开狭窄操作，例如列出资源组 X 中空闲虚拟机、创建仅演练建议，或停止一台明确获批的虚拟机。执行器身份只应拥有该契约所需的提供商操作。高风险操作需要审批、参数验证、幂等性和证据。除非明确产品用途就是受治理的访问管理，否则执行器绝不能拥有角色分配权限。

5.8 在 Azure 与中国领先云平台之间转译身份

四个平台面对相同的基本身份问题，但资源层次、策略语法、角色集成和托管身份覆盖并不相同。先转译工程意图——具名主体、临时身份验证、最小权限策略、作用域、信任、机密管理与证据——再验证实际服务契约。

表 5-7 身份与最小权限的控制平面转译。

工程意图	Azure	阿里云	腾讯云	火山引擎
员工身份	Microsoft Entra 用户/组；Azure 角色；PIM 提供资格制/限时访问。	RAM 用户/组；企业或资源目录身份功能；通过角色代入获得临时访问。	CAM 子用户/组与组织身份管理功能；使用 CAM 角色访问账户。	IAM 用户/组与组织/账户控制；验证目标区域的员工 SSO 覆盖。

工程意图	Azure	阿里云	腾讯云	火山引擎
工作负载身份	托管身份；服务主体；工作负载身份联合。	RAM 角色、ECS 实例 RAM 角色、Function Compute 角色、OIDC/RRSA 场景、STS 令牌。	CAM 角色、STS 临时凭据，以及部分 TKE 与服务场景中的 OIDC 角色集成。	IAM 角色与 AssumeRole 临时凭据；服务专用附加机制不同。
授权模型	角色定义 + 角色分配 + Azure 作用域；部分 ABAC 条件。	系统/自定义 RAM 策略、资源 ARN、条件、信任策略、资源组/账户作用域。	预设/自定义 CAM 策略、QCS 资源语法、条件、角色与信任策略；API 粒度因服务而异。	附加到用户、组或角色的系统/自定义 IAM 策略；TRN 资源语法与条件。
临时凭据	Microsoft Entra 访问令牌；托管身份令牌；联合交换。	代入角色后获得带临时 AccessKey 对与安全令牌的 STS 令牌。	代入角色后获得带临时 SecretId/SecretKey 与会话令牌的 STS 令牌。	在支持时，代入角色后获得临时 AK/SK 与令牌。
机密管理	Azure Key Vault，使用 RBAC、托管身份、私有访问和轮换集成。	KMS 凭据管理器 / 机密功能；RAM 角色访问；受支持的凭据轮换。	Secrets Manager（SSM）与 CAM、CloudAudit、轮换和 Kubernetes 场景集成。	KMS 与服务凭据功能；验证所选服务的机密管理器和工作负载集成。
证据	Entra 登录日志、Activity Log、资源/数据日志、访问复核。	ActionTrail、RAM/STS 记录、服务日志、KMS 记录。	CloudAudit、CAM/STS 记录、SSM 审计与服务日志。	根据产品支持使用 CloudTrail/审计以及 IAM/服务记录；验证数据事件覆盖范围。

5.8.1 阿里云 RAM

阿里云 RAM 区分用户、组与角色。RAM 角色没有密码或永久 AccessKey 对。受信主体代入角色后，获得包含临时 AccessKey 对和安全令牌的临时 STS 令牌。当前 RAM 指南建议编程访问使用 STS 令牌，因为凭据会自动到期，与长期 AccessKey 对相比暴露有限。[14] [15]

对阿里云上的工作负载，实例 RAM 角色和 Function Compute 角色让平台能够提供临时凭据，而无需嵌入 AccessKey。KMS 凭据管理器可以存储无法消除的凭据，并支持受控检索和部分轮换场景。权限审查必须同时检查角色信任策略与所附权限策略。[16]

5.8.2 腾讯云 CAM

腾讯云 CAM 提供用户、组、角色和策略。角色不需要永久密钥；STS API 签发临时凭据，其操作、资源、条件和有效期受策略与会话限制。腾讯文档建议子账户遵循最小权限，并把人员子账户与编程身份分开。[17] [18]

Secrets Manager（SSM）集中管理敏感值，与 CAM 集成进行访问控制，与 CloudAudit 集成提供证据，并支持机密生命周期与轮换场景。当前 TKE 指南还介绍了部分场景下用于无密码工作负载访问的 OIDC 与 CAM 角色集成。确切资源级授权粒度因服务和 API 而异；如果服务只支持操作级授权，即使策略指定了资源，也不会让操作变成资源级。

5.8.3 火山引擎 IAM

火山引擎 IAM 公开用户、用户组、角色和策略。附加到身份的策略授予其中描述的权限；角色使用信任关系，并通过角色代入获得临时凭据。火山引擎发布 IAM 策略元素与 TRN 资源格式，其访问密钥指南建议在仍需密钥时，为工作负载建立权限最低的独立用户。[20] [21]

必须逐项服务验证身份集成与资源粒度。不能因为全局 IAM 服务支持角色，就推断每个工作负载都会获得平台托管身份。应记录具体计算、函数、容器、数据库、模型或智能体服务能否在不存储密钥的情况下代入角色，适用何种令牌期限，以及由哪项审计服务记录调用。

5.9 运行身份控制平面

角色分配创建后，身份工作并未完成；还必须观测、复核和退役。生产身份计划应回答四个运维问题：存在哪些主体？哪些权限有效？哪些权限被使用？哪些权限不再合理？

表 5-8 身份运行信号与审查问题。

信号	问题	示例操作
特权分配清单	谁拥有 Owner、访问管理、策略豁免、密钥恢复或等价权限？	转为更窄角色、作用域和 PIM 资格。
未使用权限与未使用身份	主体在审查窗口内是否登录或使用过角色？	移除陈旧角色分配；检查依赖项后禁用并退役身份。
凭据年龄与类型	工作负载是否仍依赖长期机密或 AccessKey？	迁移到托管身份、联合或 STS；轮换剩余凭据。
授权失败	拒绝事件是预期负向测试、部署故障还是攻击信号？	关联主体、操作、作用域、令牌受众与变更历史。
角色与信任变更	谁更改了角色定义、信任策略、组成员关系或分配？	要求审查，对特权变更告警，保留相关性 ID。
数据平面访问	管理读取者能否通过另一个角色或密钥读取业务数据？	枚举 DataActions、服务原生 ACL 与共享密钥。
孤立主体	托管身份或服务主体是否仍附加到活动资源，并由团队拥有？	按顺序移除附加关系、分配、凭据与主体。

信号	问题	示例操作
破窗就绪性	紧急身份能否完成身份验证？使用时是否立即产生证据并触发响应？	按规程测试；保护凭据；每次使用均告警。

5.9.1 身份 SLO 与策略目标

身份治理可以表达为可度量目标。例如：在支持时，100% 生产工作负载使用托管或联合身份；源码存储库中长期云凭据为 0；100% 人员特权角色采用资格制或限时制；100% 角色分配具有所有者与用途元数据；每次策略变更都运行授权拒绝测试；孤立身份在规定退役窗口内移除。

这些目标不应变成虚荣指标。如果一个宽泛托管身份服务所有工作负载，“无机密”毫无意义；如果管理员激活的是 Owner 而不是工作职责角色，“所有管理员都使用 PIM”也不完整。身份指标必须与作用域、操作和证据质量结合。

5.10 实验 3：创建并验证只读服务身份

该独立实验验证三项概念。第一，工作负载无需客户端机密即可进行身份验证；第二，资源组作用域的 Reader 角色分配允许管理平面清单访问；第三，同一身份尝试管理写入时会被拒绝。实验使用附加到短期 Azure Container Instance 的用户分配托管身份，运行 Azure CLI 镜像。[12] [13]



图 5-6 实验把只读托管身份附加到临时容器，并同时验证允许与拒绝路径。

表 5-9 实验 3 概览。

项目	值
主要能力	工作负载身份、Azure RBAC 作用域、最小权限验证。

项目	值
Azure 资源	一个资源组、一个用户分配托管身份、一项角色分配、一个临时容器组。
成本等级	低。托管身份与 RBAC 不直接收费；Container Instances 只运行几分钟。
必需证据	身份资源 ID 与主体 ID；Reader 分配；成功资源列表；被拒绝的资源组更新；容器日志；清理结果。
设置所需权限	在实验作用域创建资源和执行 Microsoft.Authorization/roleAssignments/write 的权限，例如适当的访问管理角色。
清理边界	删除专用资源组，并验证其不再存在。

5.10.1 前提条件与安全边界

- 已安装 Azure CLI，使用 az login 完成身份验证，并指向沙盒订阅。
- 具有创建资源组、用户分配托管身份、角色分配与 Azure Container Instance 的权限。
- 支持 Azure Container Instances 与用户分配托管身份的区域。
- 带 openssl 的 shell，用于生成短后缀；也可用其他唯一值替换后缀命令。
- 不要使用生产订阅。实验会有意生成一项失败授权事件。
- 授权变更传播可能需要时间。如果容器首次尝试时无法读取资源组，请短暂等待后重新运行容器步骤，不要扩大角色。

实验只使用 Azure 管理平面。Reader 不授予 Blob、数据库、机密、搜索文档或模型推理访问。5.10.8 节提供了用于证明控制平面/数据平面区别的可选扩展。

5.10.2 设置变量并创建隔离作用域

```
suffix=$(openssl rand -hex 3)
rg="id-lab-$suffix"
location="eastus"
identity="id-reader-$suffix"
runner="id-runner-$suffix"
subscription_id=$(az account show --query id --output tsv)

az group create \
  --name "$rg" \
  --location "$location" \
  --tags purpose=identity-lab lifecycle=ephemeral owner=reader

rg_id="/subscriptions/$subscription_id/resourceGroups/$rg"
```

选择适合订阅的区域。专用资源组既是授权作用域，也是清理边界。其标签会在创建任何工作负载之前表明所有权与生命周期意图。

5.10.3 创建用户分配托管身份

```
az identity create \
  --resource-group "$rg" \
  --name "$identity"

identity_id=$(az identity show \
  --resource-group "$rg" \
  --name "$identity" \
  --query id --output tsv)

principal_id=$(az identity show \
  --resource-group "$rg" \
  --name "$identity" \
  --query principalId --output tsv)

client_id=$(az identity show \
  --resource-group "$rg" \
  --name "$identity" \
  --query clientId --output tsv)

printf "identity_id=%s\nprincipal_id=%s\nclient_id=%s\n" \
  "$identity_id" "$principal_id" "$client_id"
```

记录三个标识符。Azure 资源 ID 用于寻址托管身份资源；主体 ID 是 RBAC 分配使用的 Microsoft Entra 对象；容器请求令牌时，客户端 ID 用于选择用户分配身份。混淆这些标识符是身份验证和角色分配错误的常见原因。

5.10.4 在资源组作用域分配 Reader

```
assignment_id=$(az role assignment create \
  --assignee-object-id "$principal_id" \
  --assignee-principal-type ServicePrincipal \
  --role Reader \
  --scope "$rg_id" \
  --query id --output tsv)

echo "$assignment_id"

az role assignment list \
  --assignee-object-id "$principal_id" \
  --scope "$rg_id" \
```

```
  --include-inherited \
  --query "[].{role:roleDefinitionName,scope:scope,principal:principalId}" \
  --output table
```

主体类型为 ServicePrincipal，因为托管身份在 Microsoft Entra ID 中表示为一种特殊服务主体。分配有意限定在专用资源组，而不是订阅作用域。Reader 允许管理平面读取操作，并排除管理写入。

5.10.5 在无客户端机密的情况下运行工作负载

容器以资源附加方式获得用户分配托管身份。环境变量只携带非机密标识符。命令使用托管身份登录、列出资源，再尝试更新资源组标签。预期的写入失败会转换为实验成功退出，使容器能够保留日志。

```
command="/bin/sh -c '
  set -eu;
  az login --identity --client-id \$CLIENT_ID --output none;
  az account set --subscription \$SUBSCRIPTION_ID;
  echo ==== READ_TEST ====;
  az resource list --resource-group \$RESOURCE_GROUP \
    --query "[].{name:name,type:type}" --output table;
  echo ==== WRITE_TEST ====;
  if az group update --name \$RESOURCE_GROUP \
       --set tags.writeTest=unexpected; then
    echo UNEXPECTED_WRITE_SUCCEEDED;
    exit 42;
  else
    echo EXPECTED_WRITE_DENIED;
  fi
'"

az container create \
  --resource-group "$rg" \
  --name "$runner" \
  --image mcr.microsoft.com/azure-cli \
  --assign-identity "$identity_id" \
  --restart-policy Never \
  --os-type Linux \
  --cpu 1 \
  --memory 1 \
  --environment-variables \
      CLIENT_ID="$client_id" \
      SUBSCRIPTION_ID="$subscription_id" \
      RESOURCE_GROUP="$rg" \
  --command-line "$command"
```

Azure CLI 镜像适合控制平面实验；应用通常会使用 Azure.Identity 或 MSAL，而不是运行 az login。安全属性相同：工作负载通过附加的托管身份获得令牌，且不会向容器提供客户端机密。

5.10.6 收集允许与拒绝证据

```
for attempt in $(seq 1 60); do
  state=$(az container show \
    --resource-group "$rg" \
    --name "$runner" \
    --query "containers[0].instanceView.currentState.state" \
    --output tsv)
  [ "$state" = "Terminated" ] && break
  sleep 5
done

[ "$state" = "Terminated" ] || {
  echo "Container did not terminate within five minutes." >&2
  exit 1
}

az container logs \
  --resource-group "$rg" \
  --name "$runner"
```

日志应在 READ_TEST 下显示资源表，在 WRITE_TEST 下显示 EXPECTED_WRITE_DENIED。列表通常包括用户分配身份与容器组。确切的 AuthorizationFailed 文本可能不同，但应标识托管身份主体和被拒绝的资源组写入操作。

检查最近失败的活动事件：

```
az monitor activity-log list \
  --resource-group "$rg" \
  --offset 2h \
  --max-events 50 \
  --query "[?status.value=='Failed'].
{time:eventTimestamp,caller:caller,operation:operationName.localizedValue,status
:status.localizedValue,resource:resourceId,correlation:correlationId}" \
  --output table
```

活动日志可用性与事件详情可能滞后，或因提供程序操作而异。主要证明是工作负载日志与角色分配。如果存在失败事件，应保留调用者、操作、资源 ID、时间戳和相关性 ID 作为授权证据。

5.10.7 验证最小权限契约

表 5-10 实验室证据清单。

检查	预期结果	失败解释
托管身份存在	资源类型为 Microsoft.ManagedIdentity/userAssignedIdentities；ID 已记录。	无法寻址或附加主体。
角色分配作用域	Reader 在实验资源组这一准确作用域分配给托管身份主体。	父作用域分配可能过宽；子作用域可能不足。
凭据路径	容器使用 az login --identity --client-id；未提供密码、证书或机密。	复制凭据会违背实验目标。
正向授权	身份能够列出实验组中的管理平面资源。	检查角色传播、令牌受众、订阅与身份附加。
负向授权	资源组标签更新因授权拒绝而失败。	写入成功表示存在继承或附加角色；应调查有效访问。
问责	容器日志以及可用时的 Activity Log 标识操作与主体。	没有证据时，允许/拒绝决策难以审计。
清理	删除资源组，且 az group exists 返回 false。	残留计算资源或身份可能继续保留成本或权限。

5.10.8 可选扩展：证明 Reader 不等于数据读取

以管理员身份创建存储账户和 Blob 容器。托管身份仍只拥有资源组作用域的 Reader。从容器中，经过身份验证的命令能够读取存储账户资源属性，但 az storage container list --auth-mode login 应被拒绝，因为 Reader 没有 Blob DataActions。随后在存储账户作用域分配 Storage Blob Data Reader，等待传播后重新运行数据平面命令；此时应成功，但仍未授予任何管理写入。测试后移除数据角色。

此扩展说明权限审查为何必须同时枚举 Actions 与 DataActions，也体现一条安全排障规则：数据请求失败时，应在狭窄资源作用域添加特定数据角色，而不是把工作负载提升为 Contributor。

5.10.9 清理并验证退役

```
az group delete \
  --name "$rg" \
  --yes \
  --no-wait
```

```
az group wait --name "$rg" --deleted
az group exists --name "$rg"
```

最后一条命令应返回 false。由于托管身份、角色分配和容器组都包含在专用资源组中，资源组提供一次经过审查的退役操作。在生产系统中，应先移除分配并分离共享用户分配身份，再删除身份资源，并按照策略保留审计证据。

5.10.10 多云转译

表 5-11 按工程意图重复实验 3。

提供商	无需凭据或使用临时凭据的身份路径	只读测试	拒绝测试
Azure	把用户分配托管身份附加到受支持计算资源，并获得 Microsoft Entra 令牌。	使用 Reader 列出专用资源组作用域中的资源。	尝试管理写入，例如更新标签。
阿里云	附加实例或 Function Compute RAM 角色，或代入 RAM 角色以获得 STS 令牌。	在支持时使用作用域限定到专用资源组或资源 ARN 的只读 RAM 策略。	尝试允许操作之外的创建/更新/删除。
腾讯云	代入 CAM 角色获得 STS 凭据，或使用受支持的 OIDC 工作负载集成。	在专用项目/资源作用域使用只读 CAM 策略。	尝试策略排除的 API 操作。
火山引擎	代入 IAM 角色获得临时凭据，或在可用时使用具体服务身份集成。	在支持的最窄 TRN/项目作用域附加只读 IAM 策略。	尝试写入并捕获 IAM/审计拒绝。

应按证据类别比较结果，而不是按控制台相似性：具名主体、临时身份验证、策略或角色、确切作用域、成功读取、失败写入、审计事件和经过验证的清理。缺少托管工作负载路径或资源级授权边界，是应记录的重要提供商或服务差异。

5.11 移交给 CloudDesk FinOps 智能体

CloudDesk FinOps 智能体以后需要读取成本与资源元数据的身份、生成建议的身份，以及为获批优化操作设置的作用域狭窄执行器。本章不实现 CloudDesk 的租户注册表、跨租户隔离、响应绑定、桌面审批或自定义无服务器运行时；这些属于《无服务器 AI 智能体开发》。

可复用输出是一组受治理的能力契约：

```
identity.leastPrivilege.create(task, principalType, resourceScope, lifetime)
  -> principalId, roleOrPolicyId, assignmentId, credentialPath
  evidence: positiveTest, negativeTest, assignmentScope, reviewer

identity.permission.audit(principalId, scope)
  -> effectiveManagementActions, effectiveDataActions, inheritedAssignments,
conditions
  evidence: sourceAssignments, lastUsed, privilegedFindings, recommendedRemovals

secret.reference.resolve(secretRef, workloadIdentity)
  -> versionedSecretValue available only to the authorized workload
  evidence: vaultId, secretVersion, retrievalPrincipal, correlationId
```

未来智能体可以请求其中一种能力，但绝不能获得权限管理身份本身。CloudDesk 控制平面应通过获批工作流创建或审查权限，而工具执行则使用由此产生的狭窄身份。

5.12 关键要点

身份是云控制平面的首要边界，因为每项管理和数据操作都来自某个主体。

身份验证、授权、凭据、令牌、委派与证据是不同的工程对象。

最小权限是横跨主体、操作、作用域、条件与时间的包络，而不是一个听起来安全的角色名称。

人员管理员、部署流水线、摄取工作进程、检索应用、操作员和智能体工具执行器不应共享同一身份。

Azure RBAC 在作用域上把主体绑定到角色定义；权限向下继承，部分条件可以收窄分配。

必须分别审查管理 Actions 与数据平面 DataActions。Reader 不是通用数据读取权限。

托管身份和工作负载身份联合消除了存储式云凭据，但没有消除对狭窄角色与精确信任策略的需要。

机密管理器保护无法避免的机密，但不应成为云工作负载的默认身份验证策略。

只有当必需操作成功、禁止操作失败，并且两者都产生可问责证据时，最小权限设计才算通过验证。

多云身份比较应转译主体、信任、临时凭据、策略、作用域、机密与审计意图，同时记录非等价性。

参考资料与官方文档

资料核验于 2026 年 8 月 24 日。最终定稿时，应重新核验内置角色定义、受支持的身份集成、令牌限制、产品名称和服务专用授权粒度。

[1] NIST Computer Security Resource Center. Least Privilege glossary and NIST SP 800-53 Rev. 5, control AC-6.

[2] Microsoft Learn. What is Azure role-based access control (Azure RBAC)? Security principals, role definitions, assignments, and scope.

[3] Microsoft Learn. Understand Azure role definitions. Actions, NotActions, DataActions, and NotDataActions.

[4] Microsoft Learn. Application and service principal objects in Microsoft Entra ID.

[5] Microsoft Learn. Managed identities for Azure resources. System-assigned and user-assigned identity lifecycle and token acquisition.

[6] Microsoft Learn. Workload identity federation concepts. External OIDC trust and secretless token exchange.

[7] Microsoft Learn. Understand scope for Azure RBAC and What is Azure ABAC? Scope inheritance and role-assignment conditions.

[8] Microsoft Learn. What is Privileged Identity Management? Just-in-time, time-bound, approved, and audited privileged access.

[9] Microsoft Learn. Best practices for Azure RBAC. Remove unnecessary privilege and use narrow scopes.

[10] Microsoft Learn. Grant permission to applications to access an Azure key vault using Azure RBAC. Control-plane and data-plane separation.

[11] Microsoft Learn. Overview of Azure Policy. Relationship between RBAC authorization and compliant resource state.

[12] Microsoft Learn. How to use managed identities with Azure Container Instances.

[13] Microsoft Learn. Sign into Azure using a managed identity and Azure CLI.

[14] Alibaba Cloud Documentation. What is RAM? RAM users, groups, roles, policies, and STS tokens.

[15] Alibaba Cloud Documentation. Best practices for identity and access control. Prioritize temporary credentials and least privilege.

[16] Alibaba Cloud Documentation. KMS Credential Manager and secure KMS access using RAM roles.

[17] Tencent Cloud Documentation. CAM role and STS temporary credential concepts; AssumeRole API.

[18] Tencent Cloud Documentation. Security Practical Tutorial and access-key recommendations for least privilege and identity separation.

[19] Tencent Cloud Documentation. Secrets Manager overview, CAM integration, CloudAudit, and rotation use cases.

[20] Volcengine Documentation. IAM basic concepts and role management. Users, groups, roles, trust, and temporary credentials.

[21] Volcengine Documentation. Policy overview, resource/TRN syntax, and Access Key practice recommendations.

[22] IETF RFC 6749. The OAuth 2.0 Authorization Framework.

[23] OpenID Foundation. OpenID Connect Core 1.0 incorporating errata set 2.

第二篇 / 第 6 章

计算、弹性与可靠性

工作负载形态、故障域和容量契约如何把虚拟机转化为可替换的云资源

云虚拟机不只是带有电源按钮的远程服务器。它是一个受治理的资源图，连接着镜像、磁盘、接口、放置选择、运行状况信号、扩缩容规则、配额、恢复机制和计费承诺。本章阐述选择、扩缩、替换、保护计算资源并控制其成本所需的控制平面推理，避免把实例与其支撑的服务混为一谈。

本章内容

- 为何计算实例是托管资源契约和依赖关系图，而不是一次 CPU 与内存采购
- 如何在选择实例系列或规格前描述工作负载形态
- 硬约束、基准测试证据、区域可用性和生命周期状态如何缩小候选范围
- 为何弹性是一个包含延迟、运行状况语义、冷却和终止策略的反馈控制器
- 垂直扩缩、水平扩缩、计划扩缩和可中断容量如何解决不同问题
- 为何可靠性声明必须指明其应对的故障域
- 何时应替换、修复、恢复、还原或故障转移计算资源
- 配额、库存、放置、镜像可用性和专用硬件如何独立于价格产生容量风险
- 即用即付、Spot 容量、预留、节省计划和容量保障如何构成不同契约
- 遥测和业务上下文如何支持安全的规格优化决策
- Azure、阿里云、腾讯云和火山引擎如何用不同资源模型表达相同工程意图
- 实验 5 如何在不依赖其他实验的情况下，生成计算依赖关系图、运行时证明、可靠性声明和月度成本估算

门户让计算资源选择看起来简单得令人误解。选择区域、镜像、实例规格、磁盘，也许再选一个公共地址，然后单击“创建”。然而，每项选择都会把工作负载绑定到物理架构、故障域、网络路径、运维模型、容量池和计费契约。虚拟机资源可能几分钟内就能创建，但不合适的选择会通过操作系统镜像、数据放置、监控假设、预留承诺和恢复程序，造成延续数年的后果。

本书所依据的课程材料描述了一种“真实的虚拟机依赖关系”，其中包括订阅或账户、资源组或项目、区域和可用区、虚拟网络、子网、安全策略、镜像、规格、地址、密钥、磁盘、监控、备份和标签。材料还把 IaaS 表述为一个工程闭环：规划、配置、保护、观察、防护、恢复和优化。本章保留这一实践视角，同时将其转化为可复用的计算控制平面模型。

核心论点很简单：计算资源选择是关于工作负载形态、故障域、弹性、运维责任和成本的决策，而不只是关于 vCPU 和内存的决策。服务不会因为某台虚拟机性能强大就变得可靠。只有架构能够检测故障、维持或重新创建容量、保留正确状态，并持续满足既定服务目标时，它才具有可靠性。同样，计算资源也不会因为实例可以调整规格就具有弹性。只有受治理的控制器能够足够迅速地添加、移除或替换容量，在不引发振荡、损坏或成本失控的情况下维持预期结果，它才具有弹性。

虚拟机仍然是最清晰的参照对象，因为其依赖关系显而易见。同样的推理随后适用于 Kubernetes 节点池、机器学习训练集群、GPU 推理池、托管运行器和智能体工具工作进程。在每种情况下，平台都没有消除物理稀缺性，而是通过可编程资源契约将其显露出来。

虚拟机的计算资源图



图 6-1 虚拟机只是更大计算资源图的中心。可靠性、安全性、可观测性和成本附着于整个资源图，而非仅附着于来宾操作系统。

6.1 计算是一项托管资源契约

物理服务器主要是一台设备。云计算资源则是一项具名契约，控制平面通过它分配设备、记录期望状态、公开生命周期操作、附加策略并生成证据。来宾操作系统和应用运行在数据平面中，但实例之所以存在，是因为提供商为其维护了控制平面状态。

这一区分解释了为何可以停止、解除分配、调整规格、重新映像、重新部署虚拟机，把它移入扩缩容组，对其执行备份、用锁保护、为成本分摊添加标签，或者在磁盘遵循不同生命周期的情况下删除它。它也解释了为何关闭操作系统不一定等于释放底层计算分配。控制平面状态与来宾状态可能产生偏离。

契约要素	架构必须回答的问题	典型证据
身份与作用域	持久资源 ID、父作用域、区域、可用区、所有者和工作负载上下文是什么？	资源清单、标签、活动历史
硬件配置	请求了何种架构、vCPU、内存、加速器、本地存储和网络限制？	实例类型/规格与有效配置
软件来源	哪个镜像、代际、引导声明、扩展和补丁策略创建了节点？	镜像引用、镜像摘要/版本、部署记录
依赖关系	附加了哪些磁盘、NIC、子网、安全控制、负载均衡器、身份和机密？	资源图和依赖资源 ID
放置与故障域	哪些位置可能发生相关故障，容量如何分布？	可用区、故障域、主机组、扩缩容组配置
生命周期	实例如何创建、启动、停止、更新、替换、保护和退役？	控制平面操作和预配状态
运行状况与弹性	由谁判定实例健康、就绪、可移除或可替换？	运行状况探测、自动扩缩策略、生命周期挂钩
经济性	哪些项目按量计费，停止时哪些项目仍收费，又适用哪些承诺或折扣？	费率假设、计费状态、利用率和承诺数据

资源图之所以重要，是因为不同节点会独立发生故障和退役。删除虚拟机时，其 NIC 和磁盘是否一并删除，取决于提供商和配置的删除语义。替换实例可以保留数据磁盘，却会使本地临时磁盘失效。把工作负载移到另一个可用区，可能改变可用规格、网络路径和数据传输成本。因此，只考虑虚拟机 ID 的可靠性或清理审查是不完整的。

控制平面规则。 切勿只用友好名称描述计算资源。运维证据应包括提供商资源 ID、区域、可用区、实例类型、镜像引用、依赖资源 ID、计费模式和当前生命周期状态。

6.2 从工作负载形态而非实例目录出发

实例目录围绕提供商的硬件代际和营销系列组织；工作负载则围绕瓶颈、延迟目标、故障容忍度和负载周期组织。因此，选择过程应从工作负载证据开始，之后再把这些证据转译为提供商特定的系列。

有用的工作负载配置文件记录的远不止平均利用率。它会捕捉需求形态：CPU 是稳定还是突发；内存工作集是否可预测地容纳；存储访问是顺序还是随机；网络包速率是否比总带宽更重要；加速器是否持续占用；工作负载能否设置检查点、重启或迁移。配置文件还要记录预配不足的后果。批处理作业晚两小时完成，与交互式 API 违反 200 毫秒延迟目标，并不是等价的故障。

维度	问题	对选择的影响
CPU	持续还是突发？单线程还是并行？需要矢量指令吗？必须使用完整物理核心吗？	通用型、突发型或计算优化型系列；核心数和代际
内存	工作集大小？对缓存的敏感性？NUMA 行为？增长模式？	内存配比、内存优化型系列、纵向扩展上限
存储 I/O	随机还是顺序？IOPS、吞吐量、延迟、队列深度？持久还是临时？	磁盘类别、磁盘数量、本地 NVMe、条带化、数据放置
网络	带宽、包速率、东西向流量、延迟、加速网络、RDMA？	NIC 限制、实例系列、放置、HPC 网络结构
加速器	GPU/FPGA/其他加速器的类型、内存、互连、驱动程序和框架兼容性？	专用系列、镜像、配额、拓扑、区域库存
负载周期	连续基线、每日计划、短时突发、批处理窗口、事件驱动需求？	即用即付、计划、扩缩容组、承诺、可中断容量
状态	进程能否重启？权威状态位于何处？替换后必须保留什么？	无状态池、附加持久磁盘、复制、检查点
可靠性	服务必须承受哪个故障域，恢复目标是什么？	副本数、可用区分布、备份与故障转移设计
合规性	是否要求特定架构、机密计算功能、租户模式、位置或批准镜像？	硬性排除其他看似有吸引力的系列

通用型、计算优化型、内存优化型、存储优化型、GPU 加速型和高性能计算等常见系列标签，只有在这一抽象层级上才有用。确切系列名称、处理器代际、加速器型号和可用性变化频繁。它们应记录在当前选择记录中，而不能成为永久架构假设。

应特别谨慎对待突发型实例。它们非常适合基线较低、偶有峰值的服务、开发环境、跳板机和低流量控制服务；但当持续 CPU 需求超过基线，或性能目标无法容忍积分耗尽时，它们并不合适。评估表面上的每小时价格时，必须同时考虑积分模型和工作负载周期。

专用 AI 和 HPC 工作负载还会增加拓扑约束。只看 GPU 数量远远不够。加速器内存、主机内存、CPU 与 GPU 配比、本地存储、驱动镜像、集合通信性能、RDMA 或 InfiniBand 可用性、配额和库存都可能主导设计。对于这些工作负载，最小选择单元可能是多加速器节点或紧密放置的集群，而不是单台虚拟机。

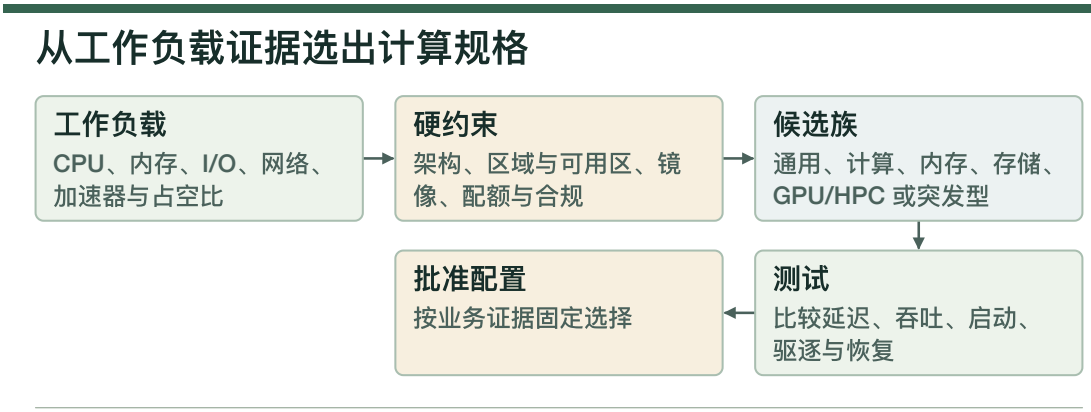


图 6-2 可辩护的计算资源选择会先排除违反硬约束的候选项，再比较价格或基准测试结果。

6.3 实例选择是一个约束满足问题

令 S 表示某个提供商公布的实例类型集合。架构应首先应用硬约束来构建可行集 F。简化表达式如下：

```
F = { s in S | architecture(s) = requiredArchitecture
             and regionAvailable(s)
             and quotaAvailable(s)
             and memory(s) >= requiredMemory
             and network(s) >= requiredNetwork
             and acceleratorCompatible(s)
             and policyApproved(s) }
```

只有确定 F 之后，团队才应使用估算成本、启动时间、能效、基准吞吐量、运维熟悉度或承诺覆盖率等软标准对候选项排序。加权评分可以支持讨论，但权重必须明确：

```
score(s) = w1 * normalizedCost(s)
         + w2 * normalizedLatency(s)
         + w3 * normalizedRecoveryTime(s)
         + w4 * normalizedOperationalRisk(s)
```

这不是为了用优化算法消除工程判断，而是为了防止诱人的价格掩盖被违反的约束。缺少所需镜像、可用区库存、加速器配额、网络功能或合规状态的候选项，不是“次优”，而是不可行。

决策记录字段	为何重要
工作负载与服务目标	把计算资源选择与结果联系起来，而不是与目录项联系起来。
硬约束	记录为何排除某些架构、区域、系列或计费模式。
候选系列与代际	防止无意中依赖某个老化或即将退役的 SKU。
基准测试方法和数据集	使性能声明可重现，并区分合成负载与代表性负载。
故障与恢复测试	说明重启、替换、检查点或可用区丢失是否可接受。
成本假设	记录时数、利用率、磁盘、许可证、网络、监控、备份和承诺覆盖率。
退出路径	说明当规格、镜像、区域或代际退役时，工作负载如何迁移。

基准测试应具有代表性、边界明确且便于比较。在启动和恢复会影响服务时，测试应包含它们。计算优化型实例即使小时费率较高，如果能更快完成作业，每项结果的成本也可能更低。相反，大型实例可能掩盖并发瓶颈，而这种瓶颈更适合通过水平分区解决。真正重要的指标往往是每项成功结果的成本，而非每小时成本。

实例生命周期也属于决策的一部分。提供商会引入新一代产品并淘汰旧产品；镜像和扩展也会老化。长期运行的工作负载需要一条升级路径，并且能在提供商强制迁移之前进行演练。记录一个系列而不是单一规格，并把镜像构建与实例创建分离，可以降低变更成本。

6.4 弹性是一个反馈回路

弹性是根据需求或策略改变已分配容量的能力。扩缩容是一项动作；弹性则是需求、测量、决策、预配、就绪和移除之间受管理的关系。一个系统即使能添加实例，如果无法在积压违反其 SLO 之前使实例就绪，那么它在技术上可扩展，在运维上却没有弹性。

垂直扩缩会改变实例容量；水平扩缩会改变实例数量。对于有状态或受许可证约束的软件，垂直扩缩通常比较直接，但可能需要停止或替换虚拟机，并最终达到系列上限。水平扩缩通常能提供更好的容错能力和增量增长，但要求工作可分区、状态外置、负载分发以

及实例安全移除。

弹性模式	触发器	优势	主要风险
手动垂直扩缩	操作员更改规格	简单，适用于已知的阶跃变化	停机、调整规格时的可用性、过度预配
手动水平扩缩	操作员更改期望数量	明确且易于推理	响应缓慢且依赖人工
计划型	日历或已知业务事件	可预测，运维成本低	预测误差和异常需求
基于指标	CPU、内存、延迟、队列长度、自定义指标	响应测得的需求	滞后、噪声、振荡、信号错误
基于事件	消息、作业、分区、请求、外部事件	直接反映排队工作	有害工作、重复事件、排空语义
预测型	预测模型估算未来需求	可在需求到来前预配	模型误差、治理、无法解释的动作

一个简单控制器可以按如下方式估算目标实例数：

```
target = clamp(minInstances,
               ceil(observedDemand / effectiveCapacityPerReadyInstance),
               maxInstances)
```

实际控制器需要考虑更多因素，包括测量窗口、启动时间、初始化、运行状况探测、连接排空、冷却、最短驻留时间、故障替换和提供商 API 延迟。它们使用迟滞机制，避免阈值附近的工作负载反复扩容和缩容。它们还区分期望数量与就绪数量。在可用区容量短缺或镜像失败期间，即使扩缩策略正确，期望容量也可能无法满足。

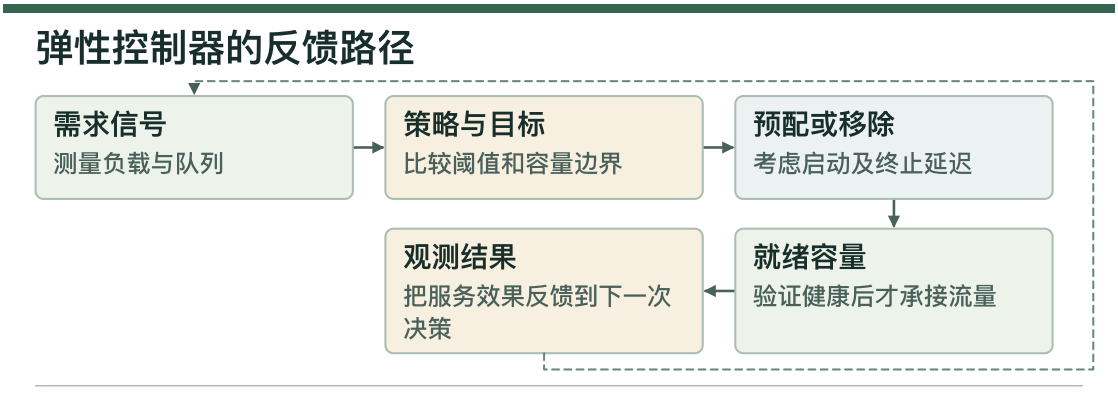


图 6-3 弹性是一个具有延迟的反馈回路。稳定运行需要明确的就绪、预热、冷却、运行状况和终止语义。

水平弹性取决于一项使节点可替换的实例契约。引导过程必须幂等；配置应在外部进行版本控制；运行状况探测应反映实例能否提供服务，而不只是进程是否存在；有状态连接和排队工作需要排空程序；数据必须位于其持久性和复制模型能够承受实例替换的系统中。

自动扩缩并没有消除容量规划。最小容量确定可靠性和延迟基线；最大容量限制成本和提供商配额暴露；默认或期望容量确定部署后的起点。因此，扩缩容组同时是可靠性对象、经济护栏和变更控制器。

弹性反模式。 仅按平均 CPU 扩缩容可能非常误导。服务的平均 CPU 可能很低，但单个分区、依赖项、锁、磁盘队列、内存限制或尾延迟路径已经饱和。

6.5 可靠性必须指明故障域

“高可用”并不是完整声明。架构必须说明它打算容忍哪些故障：进程崩溃、虚拟机故障、主机维护、机架或故障域中断、可用区中断、区域中断，或者身份、DNS、存储或部署控制平面等共享依赖项丢失。

可用性集和等效的故障域构造，可减少区域内主机或机架的相关故障。可用区通过把副本放在同一区域内物理分离的数据中心组中，处理更大的故障域。扩缩容组可以把实例分布到多个可用区或故障域，但扩缩容组本身并不能使应用具备可用区韧性。负载均衡器、状态存储、镜像来源、身份路径、DNS、监控和部署过程也都必须承受已指明的故障。

故障事件	计算控制	必须检查的其他依赖项
进程崩溃	服务监督器、重启策略、运行状况探测	启动幂等性和本地损坏
虚拟机或主机丢失	多个实例、替换控制器	负载均衡、外置状态、就绪状态
机架/故障域事件	平台分散放置或可用性集	共享网络和存储路径
可用区中断	跨可用区副本、支持可用区的扩缩容组	可用区冗余数据、负载均衡器、DNS、容量
区域中断	跨区域部署与故障转移	复制延迟、身份、流量引导、运行手册
容量短缺	多个规格/可用区、容量预留、回退策略	配额、镜像可用性、部署自动化
操作员或部署错误	版本化镜像、分阶段推出、回滚	独立证据和审批路径
共享控制平面依赖项	预先规划的降级模式和恢复路径	备用身份、DNS、机密和运维访问

对于彼此独立、完全相同且单个可用性为 a 的副本，当 n 个副本中的任意一个都能服务请求时，简化模型给出的服务可用性为 1 - (1 - a)^n。这个公式有助于建立直觉，却不能当作保证。云副本很少真正独立。它们可能共享可用区、负载均衡器、存储账户、证书、身份

提供程序、部署错误或数据损坏事件。可靠性工程在很大程度上就是发现并打破这些隐藏关联。



图 6-4 可靠性控制必须与故障域相匹配。在某一层级添加副本，并不会自动保护更高层级或共享依赖项。

设计还应区分可用性与持久性。跨可用区部署的两台无状态 Web 虚拟机可以在一个可用区不可用时继续提供服务，但并不能保护权威数据库。磁盘有三个副本，也不能保证应用级一致性。备份保护可恢复的历史，复制支持连续性，两者都不能替代对方。

6.6 替换、修复与恢复会保留不同状态

云原生指导经常倾向于替换而非修复。这一倾向很有价值，因为替换会检验声明的镜像与引导路径、减少配置漂移，并且适合自动扩缩容组。但它不是普适规则。正确响应取决于必须保留哪些状态，以及服务需要多快恢复。

工件或机制	它保留什么	它不能证明什么
镜像	操作系统和软件基线	当前数据、运行时状态或引导成功
IaC 声明	期望资源图和配置参数	来宾数据、应用正确性或提供商容量
快照	时间点块存储状态	应用一致性，除非经过协调
备份	具有保留策略、可恢复的受保护历史	立即可用性或当前状态

工件或机制	它保留什么	它不能证明什么
复制	用于连续性的当前或近当前副本	在没有历史版本时防范逻辑损坏
扩缩容组模型	可重复的实例配置和期望数量	有状态服务法定人数或数据安全性
运行手册和恢复测试	执行恢复的运维能力	未来仍会成功，除非重新测试且证据仍然有效

当权威状态位于其他位置、初始化具有确定性且服务能容忍替换间隔时，替换是合适的。当本地状态、许可配置、取证证据或较长的重建时间使替换更具破坏性时，修复是合适的。当服务或数据已越过可用性或完整性边界，必须从受保护状态还原、故障转移或重建时，则需要恢复。



图 6-5 替换、修复与恢复是不同操作，因为它们保留不同状态并满足不同恢复目标。

可靠的可替换实例设计应采用版本化镜像、声明式配置、工作负载身份、外置机密、运行状况检查、优雅终止和经过测试的预配路径。它还应记录哪些资源应随实例删除，哪些应保留。缺少这些语义时，“可替换”可能意味着“容易丢失数据”。

备份和灾难恢复应与 RPO、RTO 关联。恢复点目标表示可容忍的数据丢失量；恢复时间目标表示还原可花费多长时间。每日备份可以很好地防范意外删除，但仍可能不适用于只允许数分钟数据丢失和数分钟停机的服务。高可用、备份和灾难恢复解决的是不同时间尺度的问题。

可靠性规则。 不要把备份算作副本，不要把副本算作备份，也不要把快照当作经过测试的恢复程序。

6.7 容量、配额和放置是彼此独立的约束

即使模板、身份和预算都正确，有效部署仍可能失败。计算资源依赖物理库存。所请求的规格可能不存在于选定可用区，订阅配额可能不足，镜像可能不支持目标架构，专用加速器也可能暂时不可用。这些都是容量故障，而非语法错误。

配额表示有权请求不超过某个上限的资源；容量表示提供商在特定时间和位置满足请求的能力。预留和节省承诺主要是计费契约。容量预留、弹性保障、专用主机或提供商特定的私有容量池，是旨在提高履约确定性的不同机制。名称和保证各不相同，因此设计必须记录实际契约，而不能仅凭“预留”一词作推断。

约束	典型症状	缓解措施
订阅或项目配额	分配前部署遭拒	尽早申请配额；监控余量；使用策略批准的回退规格
区域或可用区库存	分配失败，或扩缩容组低于期望数量	使用多个可用区/规格；容量保障；分阶段推出
镜像可用性	所选镜像或代际在目标区域/架构中不可用	复制或发布批准镜像；推出前验证
专用硬件稀缺	GPU/HPC 容量不可用或等待时间长	预留容量；放宽拓扑；为作业设置检查点；将工作排队
放置约束	无法满足反亲和性、邻近性、专用主机或网络功能要求	把放置建模为硬约束，并按真实规模测试
预配时间	容量存在，但无法在需求到来前就绪	预热池、计划扩容、更小的镜像、预验证引导

容量余量应在真正有意义的层级表达。若 C_ready 是就绪服务容量，C_required 是满足服务目标所需的需求，则余量 H = C_ready - C_required。一个可用区内 H 为正并不等于具备区域韧性。vCPU 配额为正也不能证明所需加速器可用。容量仪表板需要分别展示配额、期望容量、已分配容量、就绪容量和服务吞吐量。

扩缩容组可通过允许多个可用区、规格优先级，或混合按需与可中断实例来提高履约能力。只有应用配置支持这些备选方案时，这种灵活性才有价值。混合规格会改变单实例容量和终止行为；混合架构或加速器代际会改变镜像、驱动程序、数值行为和性能。因此，回退项应是明确批准的配置，而非任意替换。

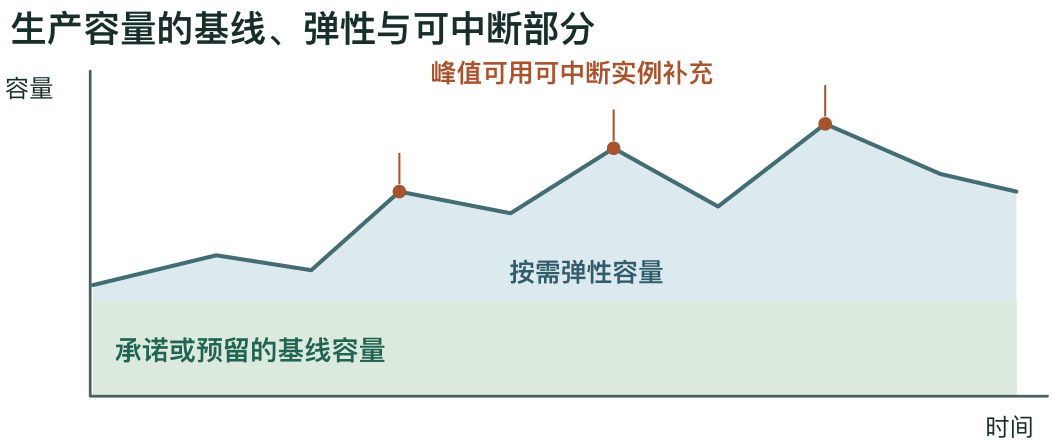
6.8 构建容量组合，而非选择单一计费方式

计算经济性是一个组合问题。不同需求部分具有不同的确定性和中断容忍度。稳定基线可以采用承诺；变化但不可或缺的需求需要按需弹性；可容错的批处理或无状态突发容量可以使用 Spot 或其他可中断市场；专用或受监管工作负载即使成本更高，也可能需要专用容量。

容量契约	最适合的场景	它不能保证什么
即用即付 / 按需	不确定、短期、持续演进或突发的需求	最低价格、未来库存或自动弹性
Spot / 可中断	可设置检查点的批处理、可丢弃工作进程、可容忍中断的分布式作业	可用性、连续性或稳定容量
预留	可预测且匹配的使用量，其中特定承诺可获得计费折扣	正在运行的实例、高可用或容量，除非契约明确包含
节省计划 / 支出承诺	不断变化、需要更广泛折扣灵活性的计算使用	无限折扣覆盖或容量保障
容量预留 / 保障	当分配风险重要时，已知的放置位置与规模	应用就绪、数据韧性或低利用率成本
专用主机 / 隔离硬件	许可证、合规性、租户或放置约束	弹性效率或自动服务可靠性

可中断容量应被视为正常运行期间就会发生的一种故障模式。工作进程必须设置检查点或保证工作幂等；队列必须安全地重新投递；扩缩容组必须替换容量；当现实比例的 Spot 实例消失时，服务仍应满足其 SLO。提供商的驱逐通知是有用的优化信号，而不是可靠性保证。

承诺应建立在测得的基线需求之上，而非乐观预测。与错误区域、系列或操作系统匹配的预留可能利用不足。节省计划虽然更灵活，仍会产生支出义务。第 14 章介绍的 FinOps 控制平面应跟踪覆盖率、利用率、到期时间，以及承诺与批准工作负载配置之间的关系。



仅把可承受驱逐和重试的工作放在可中断容量上。图示为概念曲线。

图 6-6 基线承诺、按需弹性和可中断突发解决的是不同经济性与可靠性问题。

6.9 可观测性使规格优化成为基于证据的变更

规格优化经常被表述为对低 CPU 利用率的简单响应。安全的规格优化是一项由多种信号和代表性观察窗口支撑的变更决策。它必须考虑内存、磁盘、网络、尾延迟、批处理期限、季节性峰值、恢复时间，以及工作负载在更大服务图中的位置。

证据	调整规格前的问题
CPU 分布	平均值是否掩盖了短时饱和、单线程限制或突发积分耗尽？
内存工作集	内存是稳定、增长、可回收，还是接近内存不足边界？
磁盘与本地存储	IOPS、吞吐量、延迟或队列深度是否受虚拟机规格或磁盘类别影响？
网络	更小规格是否会降低带宽、包速率、NIC 数量或加速网络支持？
应用 SLI	p95/p99 延迟、错误率、吞吐量和作业完成时间会发生什么变化？
弹性	减少大型实例，还是增加小型实例，能否改善恢复与成本？
故障行为	候选规格是否存在于所有必需可用区，并能足够快地恢复？
经济性上下文	是否包括许可证、磁盘、数据传输、承诺和运维工作？

有用的效率指标会把资源成本与交付结果联系起来，例如每个成功请求、每个已完成训练轮次或每份已处理文档的成本。利用率是输入，不是目标。利用率过高会降低韧性并增加延迟；利用率过低可能是有意设置的备用容量，也可能是资源过大的症状。目标是在可接受的单位成本下留出足以满足 SLO 的余量。

Azure Advisor 等提供商推荐系统可通过分析遥测，识别利用率不足的虚拟机和规模集。推荐是证据，而非权威。应结合业务周期、故障域要求、容量可用性和回滚就绪情况进行审查。自动执行应从只读清单和推荐能力开始，之后生成经过审查的变更计划。

计算资源状态也会影响计费。来宾操作系统关机后，资源可能仍处于已分配状态。解除分配虚拟机可以停止计算费用，但磁盘、地址、备份、监控和其他依赖项仍可能产生成本。因此，证据包应同时记录运维状态和计费状态。

6.10 Azure 参考控制平面

Azure 通过标准虚拟机资源和虚拟机规模集公开计算能力。虚拟机资源引用硬件配置、存储配置、操作系统配置、网络接口、身份、可用区、诊断和扩展。附加磁盘和 NIC 仍是可单独寻址的资源。因此，Azure Resource Graph 非常适合清点依赖关系图。

虚拟机规模集增加组级期望状态、实例编排、扩缩容、运行状况、升级和放置语义。在撰写本文时，Microsoft 建议大多数新的规模集部署使用灵活编排。灵活模式管理标准虚拟机资源，可支持混合规格以及 Spot 与按需容量组合；统一模式强调同质规模集模型。此建议和功能矩阵都是易变事实，发布或实施前应重新核实。

Azure 对象	控制平面角色	可靠性或弹性意义
Microsoft.Compute/virtualMachines	具有硬件、镜像、磁盘、NIC、身份和生命周期状态的单个计算实例	基本替换单元；本身并不是服务级高可用设计
Microsoft.Compute/virtualMachineScaleSets	组级期望状态、编排、扩缩容、运行状况、放置和升级	维持或改变容量，并可把实例分布到多个故障域
Microsoft.Compute/disks	持久操作系统或数据块存储资源	独立的数据生命周期；需要考虑快照/备份
Microsoft.Network/networkInterfaces	连接到子网、IP 配置和安全控制的数据平面附件	必须纳入替换和放置设计的依赖项
可用区 / 可用性集	放置和相关故障控制	定义副本旨在容忍哪些基础设施故障
Azure Monitor / VM Insights / 警报	遥测、运行状况、诊断和决策信号	支持故障检测、扩缩容和规格优化证据
Azure Backup / Recovery Services 保管库	受保护的恢复点和还原工作流	恢复能力，而非可用性副本
Azure Advisor / Cost Management	推荐、费率、承诺和成本证据	支持规格优化和经济组合决策

当区域和工作负载支持，且需要可用区级韧性时，应优先采用可用区。在不支持可用区的区域或场景中，可用性集仍然有用，但它应对的是更小故障域。跨可用区规模集可以把实例分布到多个可用区；应用及其数据仍然需要感知可用区的设计。

Azure Spot 虚拟机以折扣价格提供闲置容量，并可能在 Azure 需要容量时被驱逐。它们没有可用性保证，要求工作负载能够容忍中断。Azure 预留和节省计划是计费构造，具体程度与灵活性不同。二者都不应被视为高可用的证明，也不能替代容量规划。

Azure Backup 为虚拟机数据提供隔离的恢复点，而虚拟机镜像和规模集模型提供可重现性。运维团队应同时测试替换和还原。能够重新部署空虚拟机，并不能证明服务数据可恢复；能够还原磁盘，也不能证明当前应用版本能够正确启动。

6.11 按工程意图进行多云转译

Azure、阿里云、腾讯云和火山引擎都公开虚拟机资源、镜像、磁盘、可用区、自动扩缩容组、监控和多种计费模式。共享的工程意图高度一致，但运维等价性并非自动成立。实例系列、配额模型、容量保障、Spot 中断行为、关机计费、镜像目录、扩缩容组策略和可用区覆盖范围各不相同。

提供商	计算资源与依赖项	弹性与故障域	经济性与容量控制
Azure	虚拟机、镜像、托管磁盘、NIC、虚拟机扩展	虚拟机规模集、可用区、可用性集、运行状况与自动扩缩策略	即用即付、Spot 虚拟机、预留、节省计划、容量预留、Advisor
阿里云	ECS 实例、镜像、块存储、ENI、安全组	自动伸缩（ESS）、多可用区放置、弹性供应、弹性保障	包年包月、即用即付、Spot、预留实例、节省计划、容量型保障
腾讯云	CVM 实例、镜像、云硬盘、ENI、安全组	弹性伸缩组和启动配置、可用区、CLB 集成	包年包月、即用即付、Spot、预留实例折扣、配额和库存约束
火山引擎	ECS 实例、镜像、磁盘、网络接口、项目和标签	自动伸缩组、规则、冷却、生命周期挂钩、区域和可用区放置	即用即付和面向 Spot 的容量、提供商特定的扩缩容与清单行为

因此，多云计算配置文件应先记录意图，再记录提供商语法：所需架构、资源配比、加速器类型、磁盘与网络性能、故障域、扩缩模式、中断容忍度、恢复目标和经济契约。之后，每个提供商适配器把该配置文件映射到批准的实例系列，并明确报告不等价之处。

例如，“两台通用型实例”并不是可移植的可靠性设计。这两个规格可能具有不同的 CPU 代际、网络基线、磁盘限制、可用区覆盖范围、启动时间或 Spot 行为。同样，一个提供商的节省产品可能按支出广泛适用，另一个提供商的预留却匹配更窄的属性集合。转译必须保留工程意图，而不是产品标签。

6.12 实验 5——构建计算资源依赖关系图和月度成本估算

这个独立实验在 Azure 中创建一台没有公共 IP 地址的小型 Linux 虚拟机，通过控制平面命令证明实例正在运行，查询其依赖关系图，并生成有记录的月度零售成本估算。实验不声称具有高可用性，其可靠性成果是明确说明已处理和未处理的故障域。



图 6-7 只有收集了声明状态、运行时证明、依赖项清单、成本假设、可靠性范围和清理证据，实验才算完成。

6.12.1 能力目标与验收标准

能力目标是把虚拟机理解为受治理资源图，并区分运行成功、可靠性和成本证据。满足以下所有条件时，实验才算成功：

- 虚拟机及其网络和磁盘依赖项从声明状态创建在隔离的资源组中。
- 虚拟机没有公共 IP 地址和入站 Internet 规则；无需 SSH 会话。
- Azure Run Command 证明来宾操作系统已启动，并能执行范围受限的脚本。
- Azure Resource Graph 返回虚拟机规格、可用区（若有）、操作系统磁盘 ID、NIC ID、子网 ID、私有 IP 地址和标签。
- 月度估算列出计算时数、磁盘费用以及每项明确排除或不确定因素。
- 可靠性声明正确指出，单台虚拟机不具备可用区韧性，也不是服务级高可用。
- 删除资源组，最终查询证明没有实验资源残留。

6.12.2 前提条件与成本边界

项目	要求
Azure 访问权限	有权创建资源组、VNet、子网、NSG、NIC、托管磁盘和虚拟机，并运行 Resource Graph 查询。
工具	Azure CLI、Bicep、Shell，以及用于估算的 Python 3 或电子表格。
SSH 密钥	向虚拟机提供公钥，但实验不会为 SSH 创建公共路由。
区域与规格	选择订阅当前可用的区域和低成本 Linux 通用型或突发型规格。示例仅把 Standard_B1s 用作可替换的默认值。

项目	要求
成本类别	如果及时删除虚拟机，成本较低。在删除前，计算、操作系统磁盘和所有保留的依赖项可能持续产生费用。
范围之外	来宾强化、应用基准测试、多可用区架构、备份和公共入口。

易变事实。 实例可用性、配额、镜像和零售价格随区域与时间变化。请把示例规格和 API 版本视为需要验证的实验默认值，而非永久有效的书中事实。

6.12.3 声明最小资源图

新建一个目录，并把以下内容保存为 main.bicep。模板有意省略公共 IP 和入站安全规则。它启用托管启动诊断，并要求虚拟机删除时一并删除 NIC 和操作系统磁盘。完整实验仓库应固定已测试的 API 版本，并把镜像引用公开为参数。

```
param location string = resourceGroup().location
param prefix string
param vmSize string = 'Standard_B1s'
param adminUsername string = 'azureuser'
@secure()
param sshPublicKey string

var tags = {
  workload: 'compute-lab'
  environment: 'lab'
  owner: 'book-reader'
  costCenter: 'learning'
  expiresOn: '2026-08-26'
}

resource vnet 'Microsoft.Network/virtualNetworks@2024-05-01' = {
  name: '${prefix}-vnet'
  location: location
  tags: tags
  properties: {
    addressSpace: { addressPrefixes: [ '10.42.0.0/16' ] }
  }
}

resource subnet 'Microsoft.Network/virtualNetworks/subnets@2024-05-01' = {
  parent: vnet
  name: 'compute'
  properties: { addressPrefix: '10.42.1.0/24' }
}

resource nsg 'Microsoft.Network/networkSecurityGroups@2024-05-01' = {
  name: '${prefix}-nsg'
```

```
  location: location
  tags: tags
  properties: { securityRules: [] }
}

resource nic 'Microsoft.Network/networkInterfaces@2024-05-01' = {
  name: '${prefix}-nic'
  location: location
  tags: tags
  properties: {
    networkSecurityGroup: { id: nsg.id }
    ipConfigurations: [
      {
        name: 'ipconfig1'
        properties: {
          privateIPAllocationMethod: 'Dynamic'
          subnet: { id: subnet.id }
        }
      }
    ]
  }
}

resource vm 'Microsoft.Compute/virtualMachines@2024-11-01' = {
  name: '${prefix}-vm'
  location: location
  tags: tags
  properties: {
    hardwareProfile: { vmSize: vmSize }
    osProfile: {
      computerName: '${prefix}-vm'
      adminUsername: adminUsername
      linuxConfiguration: {
        disablePasswordAuthentication: true
        ssh: { publicKeys: [ { path:
'/home/${adminUsername}/.ssh/authorized_keys', keyData: sshPublicKey } ] }
      }
    }
    storageProfile: {
      imageReference: {
        publisher: 'Canonical'
        offer: 'ubuntu-24_04-lts'
        sku: 'server'
        version: 'latest'
      }
      osDisk: {
        name: '${prefix}-osdisk'
        createOption: 'FromImage'
```

```
        deleteOption: 'Delete'
        managedDisk: { storageAccountType: 'Standard_LRS' }
      }
    }
    networkProfile: {
      networkInterfaces: [ { id: nic.id, properties: { deleteOption: 'Delete' }
} ]
    }
    diagnosticsProfile: { bootDiagnostics: { enabled: true } }
  }
}

output vmId string = vm.id
output nicId string = nic.id
output osDiskName string = vm.properties.storageProfile.osDisk.name
```

部署前请验证镜像和规格。以下命令仅为示例；请选择当前在目标区域和配额中可用的镜像与规格：

```
LOCATION=eastus2
VM_SIZE=Standard_B1s

az vm image list \
  --location "$LOCATION" \
  --publisher Canonical \
  --offer ubuntu-24_04-lts \
  --all --output table

az vm list-skus \
  --location "$LOCATION" \
  --size "$VM_SIZE" \
  --resource-type virtualMachines \
  --all --output table
```

6.12.4 部署并证明运行时就绪

```
RG=book-ch06-compute
LOCATION=eastus2
PREFIX=ch06demo
SSH_KEY_FILE="$HOME/.ssh/id_ed25519.pub"

az group create --name "$RG" --location "$LOCATION" \
  --tags workload=compute-lab environment=lab owner=book-reader
costCenter=learning

az deployment group create \
```

```
  --resource-group "$RG" \
  --template-file main.bicep \
  --parameters prefix="$PREFIX" vmSize=Standard_B1s \
               sshPublicKey="$(cat "$SSH_KEY_FILE")"

az vm run-command invoke \
  --resource-group "$RG" \
  --name "${PREFIX}-vm" \
  --command-id RunShellScript \
  --scripts 'uname -a; uptime; echo compute-lab-ready' \
  --output json > runtime-proof.json
```

Run Command 是一项范围受限的控制平面操作，可以避免仅为实验创建公共管理路径。它不能替代生产访问设计。请保存完整 JSON 响应，因为它证明部署成功已经推进到来宾执行阶段。

6.12.5 查询依赖关系图

如有需要，请安装 Resource Graph 扩展，然后运行一项把虚拟机与其磁盘和 NIC 联接的查询。保存 JSON 和 CSV 两种格式。Resource Graph 架构可能演进，因此调整查询时应检查返回的属性。

```
az extension add --name resource-graph --upgrade

QUERY=$(cat <<'KQL'
let vm = Resources
| where type =~ 'microsoft.compute/virtualmachines'
| where resourceGroup =~ 'book-ch06-compute'
| extend vmSize=tostring(properties.hardwareProfile.vmSize),
         osDiskId=tostring(properties.storageProfile.osDisk.managedDisk.id),
         nicId=tostring(properties.networkProfile.networkInterfaces[0].id),
         zone=tostring(zones[0])
| project vmId=id, vmName=name, location, zone, vmSize, osDiskId, nicId, tags;
let disks = Resources
| where type =~ 'microsoft.compute/disks'
| project osDiskId=id, diskName=name,
          diskSku=tostring(sku.name), diskSizeGB=toint(properties.diskSizeGB);
let nics = Resources
| where type =~ 'microsoft.network/networkinterfaces'
| project nicId=id, nicName=name,

privateIp=tostring(properties.ipConfigurations[0].properties.privateIPAddress),

subnetId=tostring(properties.ipConfigurations[0].properties.subnet.id);
vm
| join kind=leftouter disks on osDiskId
```

```
| join kind=leftouter nics on nicId
| project vmName, location, zone, vmSize, diskName, diskSku,
          diskSizeGB, nicName, privateIp, subnetId, tags
KQL
)

az graph query --graph-query "$QUERY" --output json > compute-graph.json
az graph query --graph-query "$QUERY" --output table
```

结果应使依赖关系图清晰可见。仅有虚拟机名称不足以作为证据；输出应包含有效规格、位置、可用区（若有）、托管磁盘属性、NIC、私有地址、子网和标签。如果可用区字段为空，应说明虚拟机是区域性或非可用区资源，而不是虚构可用区韧性。

6.12.6 生成月度零售成本估算

除非工作负载计划另有说明，否则使用 730 小时作为透明的月度规划约定。通过 Azure Retail Prices API 获取当前即用即付 Linux 计算计量项，或在 Azure 定价计算器中验证。API 可能返回多个计量项，包括 Windows、Spot、低优先级或次要记录；请检查候选项，不要直接采用第一条响应。

```
REGION=eastus2
SKU=Standard_B1s

curl -sG 'https://prices.azure.com/api/retail/prices' \
  --data-urlencode "api-version=2023-01-01-preview" \
  --data-urlencode "\$filter=serviceName eq 'Virtual Machines' and armRegionName
eq '${REGION}' and armSkuName eq '${SKU}' and priceType eq 'Consumption'" \
  > retail-price-candidates.json

python - <<'PY'
import json
with open('retail-price-candidates.json') as f:
    data=json.load(f)
for x in data.get('Items', []):
    print(x.get('productName'), x.get('skuName'), x.get('meterName'),
          x.get('unitOfMeasure'), x.get('retailPrice'), x.get('currencyCode'))
PY
```

请创建工作表，而不是只给出一个数字。估算应展示假设和排除项：

明细项	数量假设	费率来源	月度估算
Linux 虚拟机计算	1 个实例 × 730 个活跃小时	当前已审查的零售计量项或定价计算器	每小时费率 × 730
托管操作系统磁盘	预配的磁盘类别和大小	当前托管磁盘价格	月度磁盘费率

明细项	数量假设	费率来源	月度估算
公共 IP	本实验中无	资源图证明不存在	0
出站数据传输	假设控制平面验证没有实质性出口流量	明确假设；未测量	实验估算中为 0
监控与日志	仅托管启动诊断；未创建 Log Analytics 工作区	明确范围	0 或当前包含的行为
备份	未启用	明确范围	0
支持、税费、协商折扣	已排除	零售估算的局限	未计入

总额是规划估算，并非账单预测。它应包括日期、币种、区域、操作系统、计量项名称、时数、磁盘类别和任何特定优惠假设。如果资源只会在上课或测试窗口运行，应同时计算计划型估算和 730 小时上限。

6.12.7 说明可靠性边界

在 reliability.md 中记录以下结论：

```
The lab deploys one VM in one region with no explicit multi-zone replica.
It can demonstrate instance creation and guest readiness, but it does not
survive VM, host, zone, or regional loss as a service. Managed disk durability
and boot diagnostics do not make the application highly available. A
production design would require multiple ready instances, load distribution,
zone-aware placement where required, externalized or replicated state, and a
tested recovery path.
```

这份声明是一项重要交付物。它可防止成功部署被错误标记为可靠架构。

6.12.8 清理并证明零残留

```
az group delete --name "$RG" --yes --no-wait
az group wait --name "$RG" --deleted

az graph query \
  --graph-query "Resources | where resourceGroup =~ '$RG' | project id, type,
name" \
  --output json > cleanup-proof.json
```

最终查询应不返回任何资源。检查订阅中是否存在创建在资源组之外的资源，并保留删除操作 ID 或活动证据。零残留验证是验收标准的一部分，因为遗留磁盘、快照、公共地址、备份保管库项或监控目标可能比虚拟机存续更久并持续产生成本。

6.12.9 证据包

工件	用途
main.bicep 与参数记录	声明的期望状态和可重现性
deployment-output.json	控制平面创建证据和资源 ID
runtime-proof.json	来宾启动和范围受限的命令执行
compute-graph.json / CSV	虚拟机、磁盘、NIC、子网、可用区和标签依赖项清单
retail-price-candidates.json	所选零售计量项的注明日期源数据
monthly-estimate.csv	透明的数量、费率、排除项和总额
reliability.md	具名故障域和明确的非保证项
cleanup-proof.json	实验资源图已不存在的证据

6.12.10 常见故障模式

故障	可能原因	正确响应
SKU 不可用	区域/可用区库存、配额、订阅限制，或规格已退役/不受支持	查询当前 SKU、检查限制、选择批准的回退项，并记录变更。
找不到镜像	优惠/SKU 已更改，或在目标区域/架构中不可用	列出当前镜像并更新参数化引用。
Run Command 失败	来宾代理尚未就绪、扩展失败，或预配仍在进行	检查实例视图和启动诊断；不要把开放公共 SSH 作为首个变通方案。
Resource Graph 联接为空	属性路径或 API 投影已更改，或资源创建尚未编入索引	分别查询每种资源类型，并检查实际属性。
价格 API 返回多行	存在多个操作系统、计量项、Spot 或节省记录，或者重复层级	检查产品、计量项、单位、区域和价格类型；与计算器交叉核对。
清理后仍有费用	依赖资源保留在组外，或软删除/受保护项持续存在	按标签和资源 ID 清点；遵循服务特定的清除或保留程序。

6.12.11 中国云平台转译

复现工程意图，而不是 Azure 语法。在阿里云上，创建一个带有镜像、系统盘、ENI/VSwitch、安全组、项目/资源组上下文和标签的小型 ECS 实例；查询 ECS 及相关资源；使用价格计算器或当前计费数据；之后释放实例并验证依赖资源。在腾讯云上，使用 CVM、云硬盘、VPC/子网、安全组、项目和标签，并记录计费模式与关机行为。在火山引擎上，使用 ECS、相关 VPC 和磁盘资源、项目与标签上下文，并使用其 Auto Scaling 或实例 API 完成清点。无论使用哪个提供商，都要说明实际故障域并证明清理完成。

6.13 CloudDesk 复用：清单、规格优化与故障域说明

CloudDesk FinOps Agent 今后可以复用本章的只读推理，而无需获得不受限制的计算管理权限。第一项能力清点完整依赖关系图；第二项根据批准的证据生成规格优化建议；第三项解释拟议架构能够和不能承受什么。

```
capability: compute.vm.inventory
risk_level: R0_read_only
inputs:
  provider: azure
  scope_id: /subscriptions/.../resourceGroups/...
outputs:
  instances:
    - resource_id
    - size
    - image
    - zone
    - disk_ids
    - nic_ids
    - billing_state
    - tags
  evidence_time
```

```
capability: compute.vm.rightsize.recommend
risk_level: R1_read_only_analysis
inputs:
  instance_id: /subscriptions/.../virtualMachines/...
  observation_window: 30d
  service_objective_id: api-prod-slo
  approved_profiles: [general-small-v2, general-medium-v2]
outputs:
  current_profile
  candidate_profile
  evidence_summary
  expected_unit_cost_change
  capacity_and_zone_constraints
  rollback_requirements
  recommendation_confidence
```

```
capability: compute.failureDomain.explain
risk_level: R0_read_only
inputs:
  workload_scope: /subscriptions/.../resourceGroups/...
outputs:
  protected_failure_domains
```

```
unprotected_failure_domains
shared_dependencies
recovery_assumptions
evidence_links
```

未来的变更型能力（如 compute.vm.resize）应单独设置。它应要求经过批准的计划、维护或滚动更新语义、并发检查、容量验证、回滚或替换策略，以及变更后的 SLO 验证。绝不能仅因为存在规格优化建议，就向智能体授予通用 Shell 或不受限制的云 CLI。

6.14 关键要点

- 云计算资源是一项具名的控制平面契约，连接着镜像、磁盘、网络接口、放置、运行状况、生命周期、证据和经济性。
- 应由工作负载形态而不是提供商目录顺序决定候选实例系列。
- 硬约束形成可行集；基准测试、恢复和成本证据用于对可行候选项排序。
- 弹性是一个具有延迟的反馈回路。期望容量、已分配容量、就绪容量和已交付服务容量是不同状态。
- 垂直、水平、计划型、基于指标、基于事件和预测型扩缩容解决不同问题，并产生不同故障模式。
- 可靠性声明必须指明故障域。多个副本仍可能共享公共依赖项和共因故障。
- 替换、修复、恢复、还原和故障转移保留不同状态，不得作为同义词使用。
- 配额、提供商库存、放置、镜像支持和专用硬件可用性是彼此独立的容量约束。
- Spot 容量、预留、节省计划和容量保障是不同契约。价格折扣不会产生高可用。
- 规格优化需要代表性遥测、应用 SLO、故障域约束和经过验证的回滚路径。
- Azure 虚拟机和虚拟机规模集资源提供参考模型；阿里云 ECS/ESS、腾讯云 CVM/AS 和火山引擎 ECS/AS 以不等价的细节实现相近意图。
- 实验 5 证明：正在运行的虚拟机、依赖关系图、成本估算、可靠性声明和清理证据是彼此独立的交付物。
- CloudDesk 可以复用范围受限的清单、推荐和解释能力，但变更计算资源需要更强的审批与验证控制。

参考资料与官方文档

[1] 美国国家标准与技术研究院：《NIST 云计算定义》。NIST SP 800-145，2011。https://doi.org/10.6028/NIST.SP.800-145

[2] Microsoft. Sizes for virtual machines in Azure. Microsoft Learn. Updated March 9, 2026. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/overview

[3] Microsoft. Availability options for Azure Virtual Machines. Microsoft Learn. Updated December 5, 2025. https://learn.microsoft.com/en-us/azure/virtual-machines/availability

[4] Microsoft. What are Azure Availability Zones? Microsoft Learn. Updated June 25, 2026. https://learn.microsoft.com/en-us/azure/reliability/availability-zones-overview

[5] Microsoft. Availability sets overview. Microsoft Learn. Updated April 6, 2026. https://learn.microsoft.com/en-us/azure/virtual-machines/availability-set-overview

[6] Microsoft. Reliability in Azure Virtual Machines. Microsoft Learn. Updated January 22, 2026. https://learn.microsoft.com/en-us/azure/reliability/reliability-virtual-machines

[7] Microsoft. Azure Virtual Machine Scale Sets overview. Microsoft Learn. Updated May 19, 2026. https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/overview

[8] Microsoft. Orchestration modes for Virtual Machine Scale Sets in Azure. Microsoft Learn. Updated May 19, 2026. https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-orchestration-modes

[9] Microsoft. Create an Azure scale set that uses availability zones. Microsoft Learn. Updated May 26, 2026. https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-use-availability-zones

[10] Microsoft. Architecture best practices for Azure Virtual Machines and scale sets. Azure Well-Architected Framework. https://learn.microsoft.com/en-us/azure/well-architected/service-guides/virtual-machines

[11] Microsoft. Azure Virtual Machines baseline architecture. Azure Architecture Center. https://learn.microsoft.com/en-us/azure/architecture/virtual-machines/baseline

[12] Microsoft. About Azure Spot Virtual Machines. Microsoft Learn. Updated February 6, 2026. https://learn.microsoft.com/en-us/azure/virtual-machines/spot-vms

[13] Microsoft. Build workloads with Azure Spot Virtual Machines. Azure Architecture Center. https://learn.microsoft.com/en-us/azure/architecture/guide/spot/spot-eviction

[14] Microsoft. Create a scale set that uses Azure Spot Virtual Machines. Microsoft Learn. Updated May 19, 2026. https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/use-spot

[15] Microsoft. Decide between a savings plan and a reservation. Microsoft Learn. Updated March 14, 2026. https://learn.microsoft.com/en-us/azure/cost-management-billing/savings-plan/decide-between-savings-plan-reservation

[16] Microsoft. What are Azure Reservations? Microsoft Learn. Updated July 17, 2026. https://learn.microsoft.com/en-us/azure/cost-management-billing/reservations/save-compute-costs-reservations

[17] Microsoft. Azure Retail Prices REST API overview. Microsoft Learn. https://learn.microsoft.com/en-us/rest/api/cost-management/retail-prices/azure-retail-prices

[18] Microsoft. Azure Resource Graph sample queries for Azure Virtual Machines. Updated August 22, 2024. https://learn.microsoft.com/en-us/azure/virtual-machines/resource-graph-samples

[19] Microsoft. Explore your Azure resources with Resource Graph. Microsoft Learn. Updated June 3, 2024. https://learn.microsoft.com/en-us/azure/governance/resource-graph/concepts/explore-resources

[20] Microsoft. Optimize virtual machines or scale sets with Azure Advisor cost recommendations. Updated February 18, 2026. https://learn.microsoft.com/en-us/azure/advisor/advisor-cost-recommendations

[21] Microsoft. Cost recommendations - Azure Advisor. https://learn.microsoft.com/en-us/azure/advisor/advisor-reference-cost-recommendations

[22] Microsoft. Reliability recommendations - Azure Advisor. Updated August 11, 2026. https://learn.microsoft.com/en-us/azure/advisor/advisor-reference-reliability-recommendations

[23] Microsoft. About Azure VM backup. Updated January 8, 2026. https://learn.microsoft.com/en-us/azure/backup/backup-azure-vms-introduction

[24] Microsoft. Back up an Azure VM from VM settings. Updated September 24, 2025. https://learn.microsoft.com/en-us/azure/backup/backup-azure-vms-first-look-arm

[25] Microsoft. States and billing status for Azure Virtual Machines. Microsoft Learn. https://learn.microsoft.com/en-us/azure/virtual-machines/states-billing

[26] Microsoft. Retired Azure VM size series. Updated July 16, 2026. https://learn.microsoft.com/en-us/azure/virtual-machines/sizes/lifecycle/retired-sizes-list

[27] Alibaba Cloud. What is Elastic Compute Service (ECS)? Updated April 15, 2026. https://www.alibabacloud.com/help/en/ecs/user-guide/what-is-ecs

[28] Alibaba Cloud. Features and specifications of ECS instance families. Updated June 25, 2026. https://www.alibabacloud.com/help/en/ecs/user-guide/overview-of-instance-families

[29] Alibaba Cloud. Benefits of Elastic Compute Service. Updated May 27, 2025. https://www.alibabacloud.com/help/en/ecs/benefits

[30] Alibaba Cloud. Create an ECS instance by using the wizard. Updated January 29, 2026. https://www.alibabacloud.com/help/en/ecs/user-guide/create-an-instance-by-using-the-wizard

[31] Alibaba Cloud. Auto Provisioning overview. https://www.alibabacloud.com/help/doc-detail/120020.htm

[32] Alibaba Cloud. Elasticity Assurance overview. Updated June 20, 2026. https://www.alibabacloud.com/help/en/ecs/user-guide/overview-of-elasticity-assurance

[33] Alibaba Cloud. Auto Scaling API overview. Updated March 13, 2026. https://www.alibabacloud.com/help/en/auto-scaling/developer-reference/api-ess-2022-02-22-overview

[34] Tencent Cloud. CVM Overview. Updated November 7, 2025. https://www.tencentcloud.com/document/product/213/495

[35] Tencent Cloud. CVM Model Selection. https://www.tencentcloud.com/document/product/213/41951

[36] Tencent Cloud. Regions and Zones. Updated February 12, 2026. https://www.tencentcloud.com/document/product/213/6091

[37] Tencent Cloud. Billing Modes for CVM. Updated May 19, 2026. https://www.tencentcloud.com/document/product/213/2180

[38] Tencent Cloud. Creating a Scaling Group. https://www.tencentcloud.com/document/product/377/8551

[39] Tencent Cloud. Creating a Launch Configuration. https://www.tencentcloud.com/document/product/377/3579

[40] Volcengine. Auto Scaling - Region and availability zone. https://www.volcengine.com/docs/6617/87001?lang=en

[41] Volcengine. CreateScalingGroup - Creates a scaling group. Updated June 26, 2026. https://docs.volcengine.com/docs/6617/109812?lang=en

[42] Volcengine. Scaling group lifecycle. https://www.volcengine.com/docs/6617/87000

[43] Volcengine. Withdraw spot instance. Updated June 18, 2026. https://docs.volcengine.com/docs/6396/155896?lang=en

[44] Volcengine. Auto Scaling SDK overview. https://www.volcengine.com/docs/6617/114423

[45] Google. Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016. https://sre.google/sre-book/table-of-contents/

[46] FinOps Foundation. FinOps Framework - Optimize Cloud Usage and Cost capability. https://www.finops.org/framework/capabilities/

第二篇 / 第 7 章

网络、私有连接与暴露控制

寻址、路由、DNS、终结点策略和证据如何把云可达性转化为受治理资源契约

云网络并不是围绕应用排列的一组子网，而是一套托管的可达性系统：名称解析为地址，路由选择路径，策略允许或拒绝流量，终结点暴露服务，证据则证明预期边界是否真实存在。因此，私有连接不是通过在资源周围画一个 VNet 实现的，而是在公共路径被有意移除或约束、私有路径可解析且获授权，并且最终状态可观察、可测试、可逆时实现的。

本章内容

- 为何云网络应建模为受治理的可达性，而非地址范围图
- 寻址、路由、分段、名称解析、终结点状态和服务策略如何组成网络路径
- 为何网络可达性与数据授权是不同决策
- 如何分别描述入口、出口、东西向、托管服务和管理平面的暴露面
- VNet/VPC、子网、路由表、安全组、NAT、负载均衡器、防火墙和边缘服务如何划分责任
- 为何专用终结点、接口终结点和网关终结点不能与服务终结点或公共防火墙规则互换
- 私有 DNS 如何成为安全和可用性边界的一部分
- 对等互连、传输路由、VPN、专线连接和全局边缘服务何时解决不同连接问题
- 有效路由、有效安全规则、连接测试、流日志和活动日志如何形成网络证据
- 为何没有公共 IP 并不能证明没有公共出口或公共服务暴露
- Azure、阿里云、腾讯云和火山引擎如何用不等价的产品表达相同网络工程意图
- 实验 6 如何证明某项托管服务可通过专用终结点访问，而其公共路径不可用
- CloudDesk FinOps Agent 今后如何复用范围受限的网络清单与暴露审计能力，而无需获得不受限制的网络管理权限

云网络通常从一套熟悉的顺序讲起：创建虚拟网络，把它划分为子网，附加安全组，再在应用前放置负载均衡器。这个顺序很有用，却可能营造一种虚假的完成感。网络图表达预期拓扑，但不能证明预期路径是否在预期 DNS 上下文中解析、遵循预期路由、通过预期策略、到达预期终结点，并从每条本应关闭的路径遭到拒绝。

控制平面视角改变了问题。它不只是问“这两个资源能否通信？”，还会问：谁声明了路径？哪个资源拥有地址？哪条路由选择了下一跃点？哪条规则允许了流量？哪个服务前门接受了它？什么证据证明结果？路径成本是多少？哪些故障域或共享依赖项会使它消失？又如何在不遗留公共终结点、过期 DNS 记录、宽松安全规则或孤立网关的情况下退役这条路径？

本章构建一个资源级云网络模型。Azure 是主要实现语言，因为其 VNet、路由、网络安全组、Private Link、Private DNS、Network Watcher 和托管服务防火墙对象使边界十分明确。对阿里云、腾讯云和火山引擎的比较依据工程意图，而非强行建立产品名称等价关系。本章最后安排一个独立实验，用同一个 Azure Blob Storage URL 完成两项测试：请求必须从公共客户端失败，并从私有虚拟机经专用终结点成功。只有 DNS、终结点状态、活动证据和清理也经过验证，实验才算完成。



图 7-1 成功的数据平面操作既需要有效网络路径，也需要有效身份和授权。这些决策彼此相关，但不能互相替代。

7.1 网络是受治理的可达性

云网络是一组关于可达性的托管声明。子网声明地址属于某个路由域；路由表声明发往目标的流量应使用某个下一跃点；安全组声明允许或拒绝选定流量；负载均衡器声明前端地址映射到健康后端；专用终结点声明某项托管服务可通过客户控制网络中的私有 IP 地址访问；私有 DNS 区域声明服务名称在受信网络内的解析结果应与网络外不同。

这些声明独立配置，也会独立出错。路由可能正确，但 DNS 指向错误地址；DNS 可解析到专用终结点，但安全组阻止端口；网络路径可能开放，但托管服务仍拒绝请求，因为公共访问已禁用、终结点连接未获批准或调用者缺少数据平面授权。反过来，调用者也可能持有有效令牌，却没有到终结点的路由。因此，可靠运行需要组合模型，而不是一个“已连接”标志。

```
networkReachable(source, name, protocol, port) =
    nameResolves(name)
    and routeExists(source, resolvedAddress(name))
    and segmentPolicyAllows(source, resolvedAddress(name), protocol, port)
    and endpointIsReady(resolvedAddress(name))
    and serviceFrontDoorAllows(source)

dataOperationAllowed =
    networkReachable
    and identityAuthenticated
    and dataAuthorizationAllows
```

这不是要作为字面代码执行，而是一份架构检查清单，用于防止不同类别混为一谈。“私有”描述服务的访问路径，并不会自动说明调用者、加密契约、授权策略、数据分类或服务级目标。

控制平面规则。 把每条必需路径视为肯定断言，把每条禁止路径视为否定断言。“应用可以私下访问数据库”与“无法通过数据库的公共终结点访问它”是两个独立验收标准。

决策	控制平面对象	证据
从名称到地址	公共 DNS、私有 DNS 区域、解析器、记录或别名	来自预期客户端上下文的查询结果；区域链接和记录清单
从地址到下一跃点	系统路由、路由表、对等互连、传输路由、网关播发	有效路由、下一跃点测试、路由传播状态
流量获准	NSG/安全组、ACL、防火墙、服务策略	有效规则、拒绝/允许连接测试、流日志
终结点已接受	公共 IP、负载均衡器前端、专用终结点、终结点服务	预配与连接状态；前端/侦听器配置
托管服务准入	公共网络访问、服务防火墙、网络安全边界、终结点策略	服务网络设置和失败的公共路径请求
数据操作获授权	托管身份、角色分配、资源策略、令牌作用域	身份验证日志、授权决策、成功的范围受限操作

7.2 网络既是资源图，也是策略图

虚拟网络通过具有不同生命周期边界的资源图实现。VNet 或 VPC 拥有地址空间；子网分配地址空间的一部分；网络接口获得私有地址；路由表和安全组附加到子网或接口；公共 IP 地址、NAT 网关、负载均衡器、防火墙、专用终结点、DNS 区域、VPN 网关、专线网关和传输路由器则增加新的路径决策与依赖关系。

该图同时也是策略图。附加到子网的 NSG 可以应用于该子网中当前及未来的所有接口；用户定义路由可以把全部默认流量重定向到防火墙设备；链接到多个虚拟网络的私有 DNS 区域可以改变多个工作负载的服务解析；对等互连可以让地址空间可达，却不会自动使关系具有传递性；公共 IP 可以暴露前端，同时让后端保持私有。因此，网络变更具有无法仅从被改资源看出的影响范围。



图 7-2 云网络同时是资源图、策略图和证据图。每个节点都有自己的身份、生命周期、成本和故障模式。

资源级网络决策记录至少应列出以下内容：

字段	问题
信任区	哪些来源和目标属于同一信任区，哪些跨区访问需要明确检查或审批？
地址所有权	谁分配每个 CIDR 块，并防止环境、区域、租户和本地网络之间发生重叠？
入口路径	哪些公共或私有前端在何种协议和端口上为哪些应用接受传入流量？
出口路径	哪些子网可以通过何种 NAT、防火墙、代理或网关发起公共或跨网络流量？
东西向路径	哪些工作负载、子网、VNet、VPC 和集群可以横向通信？
托管服务访问	哪些 PaaS 服务使用公共终结点、服务终结点、专用终结点或服务特定的私有访问？
DNS 上下文	每个客户端位置由哪个解析器和区域决定每个服务名称对应的地址？
检查与证据	在哪里观察有效规则、路由状态、流量记录、运行状况和控制平面变更？
经济性	哪些路径会产生网关、终结点、处理、跨可用区、跨区域或 Internet 传输费用？

字段	问题
退役	必须删除或分离什么，以及按何种顺序才能彻底移除路径?

7.3 寻址、分段与名称解析

地址规划是一项伪装成设置表单的长期架构决策。VNet 或 VPC 地址空间不仅要容纳今天的实例，还必须与对等网络、本地范围、托管服务集成、Kubernetes Pod 和 Service 范围、收购网络、灾难恢复区域和未来租户分区共存。重叠不只是给路由带来不便，还会使预期连接变得含糊甚至不可能，除非使用转换、代理或重新设计。

分段应遵循信任和运维边界，而非任意的数值对称。三层应用不一定需要三个子网，三个子网也不会自动形成三个安全区。只有分段具有具名用途、地址所有权、路由策略、安全策略、生命周期所有者、证据和容量计划时，它才有意义。用于专用终结点、委派服务、网关、防火墙或 Kubernetes 节点的子网还可能有服务特定约束。应在地址范围被占用前记录这些约束。

云安全组通常有状态，但不同提供商的附加和评估模型各异。网络 ACL 可能无状态或面向子网；防火墙策略可能检查应用名称或威胁情报，而安全组通常作用于地址、服务标签、协议和端口。由于这些控制回答的问题不同，增加更多控制并不会自动形成纵深防御，反而可能产生互相矛盾的策略和不明确的所有权。

DNS 是路径的一部分，而非便利层。私有连接通常依赖分割视野行为：应用继续使用服务的普通完全限定域名，但受信解析器返回专用终结点地址；外部解析器继续返回公共服务路径。这样可以保留应用配置，同时根据客户端上下文改变网络目标。它也意味着解析器可用性、区域链接、转发规则、记录生命周期和缓存行为都成为服务可靠性的一部分。

控制平面规则。 不要用硬编码私有 IP 替换托管服务 FQDN。终结点地址和服务记录都是受控资源。应保留提供商支持的名称，并控制解析上下文。

7.4 路由、传输与显式出口

路由选择路径，但不授予信任。云平台通常为本地子网可达性、更广的 VNet 或 VPC 地址空间以及默认目标创建系统路由。用户定义路由可以覆盖选定的系统路径，把流量引向虚拟设备、VPN 网关、专线网关或其他下一跃点。通过 BGP 学习的动态路由可在混合拓扑发生变化时添加或撤销前缀。因此，网络接口看到的有效路由是系统路由、用户定义路由、传播路由、对等互连和服务特定行为共同作用的结果。

对等互连为两个虚拟网络提供直接连接，但直接并不等于可传递。若分支 A 与中心对等，分支 B 也与同一中心对等，A 和 B 并不会自动通过中心通信。传输需要明确的路由或网关设计。Azure Virtual WAN、带传输路由器的阿里云 CEN、腾讯云 CCN 以及火山引擎

CEN 或传输路由器能力，以不同规模产品化了这一职责。其工程意图都是集中路由传播和连接策略，具体资源与计费模型则由提供商决定。

出口需要与入口同等严谨的设计。私有虚拟机仍可通过默认出站访问、实例公共 IP、负载均衡器出站规则、NAT 网关、防火墙或代理发起 Internet 连接。因此，“没有公共 IP”并不能证明“没有公共出口”。架构应明确出口所有者、公共源地址、允许的目标、检查点、DNS 策略、端口容量、日志路径和故障行为。

Azure 当前指导明确体现了这种转变：新虚拟网络默认使用私有子网，需要公共访问的工作负载应采用 NAT Gateway、Azure Firewall 或代理等显式出站方法。确切推出安排和产品行为属于易变事实，但稳定的工程原则不变：隐式出口会产生隐藏依赖、不稳定源地址和不清晰证据；显式出口则把意外路径转化为具名资源契约。

控制平面规则。 易变平台事实。Azure 网络默认值、SKU 可用性和出站优先级规则会演进。依赖实验默认值或迁移顺序前，请验证当前区域行为。

出口机制	它提供什么	它不提供什么
实例公共 **IP**	受安全策略约束的直接地址和双向 Internet 路径	集中源地址、集中检查或共享生命周期
NAT 网关	子网级源地址转换和可预测出站 IP	未经请求的入站访问或应用感知检查
防火墙 / 安全 **Web** 网关	集中策略、检查、日志记录和受控出口	自动应用授权或 DNS 正确性
负载均衡器出站规则	与负载均衡器前端关联的出站 SNAT	通用出口治理，或防范所有 SNAT 耗尽模式
代理	应用协议中介、目标策略，以及可能的身份感知访问	透明支持每种协议或路径
无显式出口	在平台真正禁用隐式访问时减少公共可达性	证明每项托管依赖和更新路径仍能工作

7.5 暴露面不止一个方向

暴露经常被简化为公共入口：应用是否有公共 IP 地址或面向 Internet 的负载均衡器。但这只是一个表面。云工作负载可能通过出站回调、横向路由、对等网络、托管服务公共终结点、DNS 重绑定、管理 API、第三方连接器，或能够创建新网络资源的智能体工具暴露。因此，审查必须至少区分五个方向。



图 7-3 暴露面包括公共入口、公共出口、东西向可达性、托管服务访问和控制平面权限。关闭一个表面不会关闭其他表面。

暴露面	典型资源	所需证据
公共入口	公共 IP、公共负载均衡器、Application Gateway、API 网关、Front Door/CDN、WAF	侦听器和前端清单；DNS；有效入站规则；成功与被拒的探测
公共出口	NAT 网关、防火墙、代理、实例公共 IP、负载均衡器出站规则	有效路由；源 IP 证明；目标策略；会话/流量证据
东西向	子网路由、对等互连、传输路由器、服务网格、内部负载均衡器	双方的有效路由与安全规则；延迟和可达性测试
托管服务前门	公共终结点、服务防火墙、服务终结点、专用终结点、终结点策略	公共网络设置；终结点状态；私有 DNS 结果；数据平面拒绝/允许证据
管理平面	门户、CLI、ARM/API、IaC 流水线、委派角色、智能体工具	角色与作用域清单；活动日志；策略；变更审批和回滚证据

暴露控制还必须区分侦听与可达，以及可达与获授权。公共前端可以存在，却没有允许的网络安全规则；内部侦听器可能从很大的对等地址空间可达；托管服务可以保留公共 DNS 名称，同时拒绝所有公共请求；即使当前数据平面是私有的，管理 API 仍可创建公共终结点。证据应记录每一层，而不是给整个应用贴上一个二元“私有”标签。

7.6 入口：第 4 层、第 7 层与全局边缘

入口资源实现不同契约。第 4 层负载均衡器根据前端地址、规则、运行状况探测和后端池分发 TCP 或 UDP 流量，适用于网络协议应以最少应用解释直接通过的场景。应用网关或应用负载均衡器理解主机名、URL 路径、标头、Cookie、证书和 Web 应用防火墙规则等 HTTP 属性。全局边缘服务则增加任播或基于 DNS 的分发、CDN 行为、全局运行状况评估和边缘安全。

这些服务可以组合，而非相互排斥。全局边缘可以路由到区域应用网关，再由后者路由到私有后端。内部第 4 层负载均衡器可以作为数据库代理或服务网格网关的前端。Private Link 可保护全局或区域前端之后的源站。设计应明确每个终止点与信任转换：TLS 在何处终止，客户端身份在何处检查，WAF 策略在何处应用，源信息如何保留，运行状况探测与真实用户请求有何不同。

在单个后端前放置负载均衡器并不会产生高可用。前端、运行状况探测、DNS、证书、后端池、可用区分布和应用状态都必须支持预期故障域。同样，边缘服务可以改善全局路由并吸收某些攻击，但无法修复损坏的区域依赖项，也无法修复错误报告运行状况的应用。

意图	典型 Azure 参照	关键设计问题
区域 **TCP/UDP** 分发	Standard Load Balancer	哪些端口和流量需要透传式第 4 层均衡？
区域 **HTTP/S** 路由与 **WAF**	Application Gateway v2 / WAF_v2	TLS、主机/路径路由和 Web 策略在哪里终止？
全局 **HTTP/S** 交付与边缘安全	Azure Front Door	如何治理全局运行状况、源站隐私、缓存和故障转移？
基于 **DNS** 的全局路由	Traffic Manager	仅在 DNS 层选择终结点而不代理流量，是否足够？
集中网络检查	Azure Firewall 或网络虚拟设备	哪些入口与出口流量需要有状态或应用感知检查？

7.7 私有连接是一项终结点契约

托管服务的私有连接经常被描述为“把服务放进 VNet”。这种说法通常并不准确。服务仍运行在提供商的服务基础设施中。专用终结点会创建一个客户可见的网络接口和私有地址，把流量通过提供商私有网络映射到特定服务资源或子资源。客户控制终结点子网、DNS 集成、受支持时的网络策略和使用者侧生命周期；提供商控制托管服务与 Private Link 实现。

此模型不同于服务终结点。服务终结点把子网身份扩展到托管服务的公共终结点，使服务防火墙可以准入该子网。即使流量可能保留在提供商骨干网上，目标仍是公共服务地址。专用终结点则把客户端看到的目标改为 VNet 或 VPC 中的私有地址。它可以支持更强的防数据外泄控制和更明确的私有 DNS，但会增加终结点、DNS、审批、配额、成本和生命周期责任。

阿里云也区分网关终结点和 PrivateLink 接口终结点。网关终结点可为受支持云服务添加路由并应用终结点策略，而不分配接口地址。PrivateLink 接口终结点使用私有网络接口和安全组，可参与更复杂的 VPC、CEN、Express Connect 或 VPN 设计。腾讯云 Private Link 和火山引擎 PrivateLink 同样把终结点使用者连接到受支持的终结点服务，但受支持后端、区域、跨账户行为和 DNS 集成各不相同。

私有访问模式	客户端看到的地址	主要策略点	典型优势	重要限制
公共终结点 + IP 防火墙	公共 IP	托管服务防火墙	对已知公共来源进行简单限制	仍使用公共终结点；源 IP 可能不稳定
服务终结点 / VNet 规则	公共服务 IP	子网身份加服务防火墙	提供商骨干网直达路径与简单服务限制	不等同于私有 IP 或 Private Link
专用终结点 / 接口终结点	使用者 VNet/VPC 中的私有 IP	终结点、安全策略、DNS 与服务连接状态	强私有路径和明确终结点生命周期	成本、配额、DNS 复杂性、服务支持限制
网关终结点	通过终结点路由的提供商前缀	路由表和终结点策略	高效访问受支持对象服务的私有路径	服务与路由支持因提供商而异
自托管私有服务	内部负载均衡器或终结点服务后的私有 IP	使用者和提供商 VPC 策略	完全控制应用服务发布	客户负责后端可用性和补丁

控制平面规则。 专用终结点不会禁用公共终结点。请单独配置托管服务的公共网络策略，并从外部客户端证明否定路径。

7.8 DNS 是安全和可用性边界的一部分

最稳健的专用终结点模式会保留服务的标准 FQDN。公共 DNS 发布一个标识服务私有链接域的规范链；链接到受信 VNet 的私有 DNS 区域用终结点私有地址覆盖最终答案。外部解析器沿公共链到达公共服务前门，而公共访问禁用时，该前门应拒绝请求。内部解析器返回私有地址，客户端由此到达专用终结点。



图 7-4 私有 DNS 会改变返回给受信客户端的地址。公共网络策略仍须关闭公共路径，数据平面授权仍须批准操作。

DNS 故障模式在运维上意义重大。区域可能存在但未链接到客户端 VNet；自定义 DNS 服务器可能未转发私有链接区域；对等 VNet 可能缺少区域链接；过期记录可能指向已删除终结点；两个独立管理的同名区域可能包含不同记录；混合客户端可能查询不向云解析器转发的本地 DNS。由于应用仍使用熟悉主机名，如果不明确捕获解析路径，这些故障可能看似普通的连接或 TLS 错误。

DNS 解析与访问控制彼此独立。公共 DNS 解析可以显示某个全局唯一服务名称存在，而不会泄露其私有地址或授予访问权限。反过来，正确的私有地址仍可能被路由、NSG、终结点策略、服务防火墙、身份或数据授权阻止。本章后续实验会把 DNS 输出与连接及服务证据一并记录，使故障域可见。

7.9 对等互连、传输、混合连接与边缘

单个 VNet 或 VPC 是区域隔离域。更大的系统会有意增加连接域。对等互连直接连接地址空间，通常针对低延迟私有流量优化；传输平台集中路由传播、分段、共享服务和多区域连接；VPN 网关通过公共网络加密流量；专线服务与企业或运营商网络建立私有线路和 BGP 关系；边缘及 CDN 服务把公共应用入口点移近用户并保护源站。

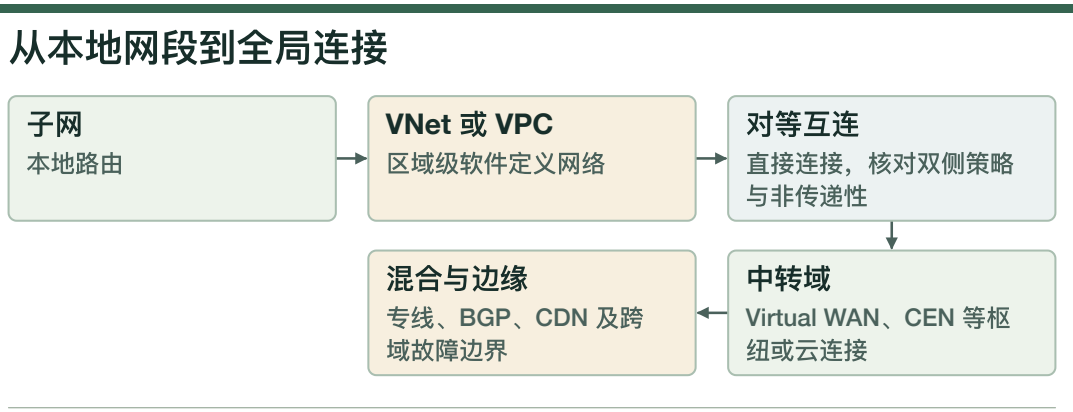


图 7-5 连接从子网扩展到区域 VNet、直接对等互连、传输域以及混合或全局边缘服务。每一步都会增加路由状态、策略、成本和故障模式。

决策不能孤立地简化为“对等互连还是传输”。它取决于网络数量、路由规模、集中检查需求、跨区域拓扑、重叠地址、共享 DNS、组织所有权和所需故障隔离。一小对 VNet 可以使用直接对等互连；大型组织可以使用中心辐射型架构或托管传输服务；混合环境可以在中心终止 VPN 或专线连接，并把路由分发到分支；全局应用可以把服务到服务流量的私有传输与用户流量的公共边缘服务组合起来。

连接并不自动意味着策略对称。双方可能都需要安全规则；路由传播可能不对称；防火墙可能看到去程路径却看不到返回路径；网络地址转换可能隐藏原始来源；对等互连和传输费用可能双向收取。设计记录应标识路径对称性、来源保留、最大容忍延迟、加密契约、路由所有权、BGP 播发、故障转移行为和诊断证据。

7.10 网络安全控制分层但不可互换

当每项控制职责明确时，网络安全最为强健。安全组和 NSG 在靠近子网或接口的位置提供分布式有状态筛选；网络 ACL 在公开此功能的提供商中提供面向子网的筛选；防火墙集中有状态或应用感知检查；Web 应用防火墙检查 HTTP/S 请求中的应用攻击；DDoS 服务保护网络或应用入口点免受流量型和协议型攻击；专用终结点移除通往受支持服务的公共网络路径；身份与授权决定谁可以执行数据操作。

这些层不能在表述上互相替代。WAF 不保护 SSH 侦听器；安全组不验证 OAuth 令牌；专用终结点不能阻止权限过大的身份删除数据；防火墙不能修复错误 DNS 区域；网络安全边界或服务防火墙不会在使用者 VNet 中创建终结点。设计应说明每项控制应对的故障或威胁类别，以及用于证明它的证据。

控制	主要层级	防范 / 治理对象	不能证明
NSG / 安全组	L3/L4 分布式策略	地址、协议、端口和服务标签可达性	应用身份、内容安全或 DNS 正确性
路由表	路径选择	下一跃点选择和流量引导	信任、授权或应用运行状况
NAT 网关	地址转换	可预测出站身份和 SNAT 容量	检查或未经请求的入站访问
防火墙	集中式 L3-L7 策略	有状态筛选、出口策略、威胁情报、检查	业务授权或后端正确性
WAF	HTTP 应用边缘	常见 Web 攻击、机器人、请求策略	非 HTTP 协议或内部横向移动
专用终结点	托管服务访问路径	私有目标地址和提供商骨干网路径	公共终结点关闭或数据授权
DDoS 防护	网络/应用可用性	服务范围内的流量型和协议型攻击	应用正确性或凭据滥用
身份与 **RBAC**	控制/数据授权	谁能管理或使用资源	网络路径存在

零信任并不意味着移除网络，而是网络位置不会被视为充分信任。身份、设备、工作负载、数据和风险信号仍然重要；最小权限网络路径降低攻击面和数据外泄机会，身份与应用策略则决定操作。因此，最实用的设计是“路径私有、策略明确、身份授权、遥测举证”。

7.11 网络证据：有效状态、测试与流量记录

只有当预期状态可以与有效状态比较时，网络控制平面才值得信任。声明的路由表可能不含系统路由或传播路由；仓库中的 NSG 文件可能不是附加到活动子网的 NSG；专用终结点可能已预配但仍待批准；DNS 区域可能包含正确记录却未链接到客户端网络。因此，诊断证据必须从与被测试声明相同的来源、目标、名称、协议、端口和时间窗口收集。

Azure Network Watcher 提供 IP 流量验证、NSG 诊断、下一跃点分析、有效安全规则、连接故障排除、数据包捕获和 VPN 故障排除工具。连接故障排除可以测试虚拟机或其他受支持来源到 FQDN、URI、IP 地址或另一资源的连接，并报告 DNS、路由、NSG、套接字和延迟信息。虚拟网络流日志在 VNet 作用域记录第 4 层流量，并取代较旧的 NSG 流日志模型。重要的架构要点不在于具体工具名称，而在于配置证据、主动测试证据和观测流量证据回答不同问题。

控制平面规则。 易变平台事实。Azure 已不再允许新建 NSG 流日志，并计划于 2027 年 9 月 30 日停用该功能。新设计应使用虚拟网络流日志，并验证受支持工作负载与区域可用性。



图 7-6 暴露控制是一个受管理的反馈回路：声明、部署、测试、观察、决策和修复。拓扑图只是第一步。

证据类型	回答的问题	典型限制
资源清单	当前存在哪些网络资源和附件？	不能证明活动数据包路径
有效路由	来源会使用哪个下一跃点到达目标？	不能证明目标接受流量
有效安全规则	哪个分布式策略应用于接口或子网？	可能不包含服务前门或应用策略
来自客户端上下文的 DNS 查询	此客户端实际收到哪个地址？	不能证明地址可达或获授权
连接测试	此来源目前能否到达此目标和端口？	时间点探测；可能不能代表真实流量规模
流日志 / 会话日志	观察到、允许、拒绝或转换了哪些流量？	保留期、采样、受支持工作负载和成本各异
活动或审计日志	谁在何时更改了控制平面资源？	通常不包含应用数据平面事件
应用跟踪	请求是否到达应用并完成？	可能无法确定具体哪项网络控制失败

7.12 网络路径的可靠性与经济性

每条网络路径都有成本和故障域。公共出口可能产生数据传输费、NAT 处理费、防火墙处理费、代理容量费和目标端费用。专用终结点通常按小时和数据处理收费。对等互连、传输、跨可用区、跨区域、专线连接和全局边缘服务分别采用不同的入口、出口、端口、网关或带宽计费模型。流日志和流量分析还会增加摄取、存储和分析成本。因此，网络成本应纳入资源图，而不是在账单中突然出现。

更便宜的路径也可能具有不同可靠性。单个虚拟设备可能成为瓶颈或共享故障域；区域性 NAT 网关可能无法满足可用区故障目标；专用终结点可能健康，但私有 DNS 不可用；VPN 可以提供加密连接，却不具备专用线路的延迟或可用性特征；专用线路可能发生故障，而路由传播或客户本地设备才是真正的恢复瓶颈。可靠性声明应列出路径、每项依赖、受保护故障域和故障转移机制。

网络优化应以证据为驱动。只有在证明所有权和依赖关系后，才能移除未使用公共 IP、空闲负载均衡器、过期对等互连、不需要的专用终结点、重复流日志和昂贵的跨区域路径。集中出口可以降低成本并改善控制，但也可能扩大影响范围。把服务移到 Private Link 后面可以减少公共暴露，却会增加终结点和 DNS 成本。优化目标不是最低网络账单，而是在可解释成本下满足所需可达性、安全性、性能和恢复结果。

7.13 按工程意图进行多云转译

Azure、阿里云、腾讯云和火山引擎共享相同的主要网络意图：隔离地址空间、分段工作负载、选择路由、控制入口和出口、连接网络、私下发布服务、解析私有名称、检查流量并收集证据。各产品在运维上并不相同。必须针对每个提供商和区域验证受支持的终结点服务、路由传播、安全组语义、私有 DNS 集成、跨区域行为、计费、配额和诊断工具。

工程意图	Azure	阿里云	腾讯云	火山引擎
区域私有网络	虚拟网络（VNet）	VPC + vSwitch	VPC + 子网	VPC + 子网
分布式数据包策略	NSG / ASG	安全组 / 网络 ACL	安全组 / 网络 ACL	安全组 / 网络 ACL
自定义路由与出口	路由表、NAT Gateway、Firewall	路由表、Internet/VPC NAT Gateway、Cloud Firewall	路由表、NAT Gateway、CFW	路由表、NAT Gateway、Cloud Firewall
L4 / L7 入口	Load Balancer / Application Gateway	NLB、CLB、ALB	CLB / ALB	CLB / ALB
私有服务访问	Private Link / 专用终结点；服务终结点	PrivateLink 接口终结点；受支持服务的网关终结点	Private Link 终结点和终结点服务	PrivateLink 终结点和终结点服务
网络互连	VNet 对等互连、Virtual WAN	VPC 对等互连、CEN / 传输路由器	VPC 对等互连、CCN	VPC 对等互连、CEN / 传输路由器
混合连接	VPN Gateway、ExpressRoute	VPN Gateway、Express Connect	VPN Connection、Direct Connect	VPN Connection、Direct Connect
私有 **DNS**	Azure Private DNS / Private Resolver	PrivateZone / DNS 解析器能力	Private DNS	PrivateZone / DNS 解析器能力

工程意图	Azure	阿里云	腾讯云	火山引擎
流量证据	Network Watcher、VNet 流日志、Activity Log	VPC 流日志、CloudMonitor、ActionTrail	传送到 CLS 的流日志、诊断、CloudAudit	VPC 流日志、CloudMonitor、CloudTrail

可移植设计记录应先描述意图，例如“工作负载子网没有隐式 Internet 出口”“托管存储只能通过私有地址访问”“公共 HTTP 通过带 WAF 的 L7 前端进入”，或“通过冗余专用线路学习本地路由”。之后，再从功能、运维、治理、经济性和故障域方面评估提供商映射是否等价。名称相似的产品仍可能需要不同架构。

7.14 实验 6——创建私有连接并证明公共暴露已禁用

这个独立实验创建一个 Azure Storage 账户、Blob 服务专用终结点、私有 DNS 区域和一台没有公共 IP 地址的小型 Linux 虚拟机。存储账户最初启用公共网络访问，只为上传一个测试对象；之后更新同一 Bicep 部署以禁用公共终结点。读者使用同一个 Blob URL 完成两项测试：从公共 Shell 发出的请求必须失败，而在 VNet 内通过 Azure Run Command 从私有虚拟机执行的请求必须解析到私有 IP 地址并返回对象。



图 7-7 实验 6 使用一个 Blob Storage FQDN 证明两条不同路径：公共路径遭拒，私有路径解析到终结点 NIC 并成功访问。

7.14.1 能力目标与验收标准

能力目标不只是创建专用终结点，而是生成一份边界明确的证据包，证明预期可达性和预期不可达性。满足以下所有条件时，实验才算成功：

- 从声明状态在隔离资源组中创建 VNet、两个子网、NSG、虚拟机、私有 DNS 区域、专用终结点、Blob 容器和存储账户。
- 虚拟机没有公共 IP 地址、入站管理规则和隐式默认出站路径。

- 专用终结点连接状态已获批准，且目标为 Blob 子资源。
- 从私有虚拟机看，普通存储 FQDN 解析为专用终结点子网中的 RFC 1918 地址。
- 同一私有虚拟机通过 HTTPS 成功获取测试对象。
- 禁用公共网络访问后，从公共 Shell 访问同一对象 URL 不会返回该对象。
- 保存存储网络设置、私有 DNS 区域链接、终结点连接、Resource Graph 清单和 Activity Log 证据。
- 删除资源组，最终查询证明没有实验资源残留。

7.14.2 前提条件与成本边界

要求	实验选择
Azure 访问权限	所在订阅允许读者创建资源组、网络资源、小型虚拟机、存储账户、专用终结点、私有 DNS 区域，并执行不依赖特定角色的 Run Command 操作。
本地工具	Azure CLI、Bicep 支持、curl、SSH 公钥，以及可保存 JSON 和文本证据的 Shell。
区域	选择一个支持所选虚拟机规格、Storage、Private Link 和 Run Command 的区域。验证当前配额和 API 版本。
成本类别	成本较低但并非免费。虚拟机、专用终结点、存储以及任何可选流日志都会产生费用。请迅速完成并删除实验。
安全边界	只使用合成内容。SAS 令牌有效期很短，不得提交到源代码管理或证据存档。
证据目录	部署前创建 evidence/，并存储命令输出、标头、DNS 输出、Resource Graph 结果和清理证明。

控制平面规则。 示例 API 版本、区域、虚拟机规格、镜像和命令输出都是需要验证的实现默认值，而非本书永久有效的事实。

7.14.3 声明私有访问资源图

把以下内容保存为 main.bicep。模板将 publicNetworkAccess 作为参数，以便先用引导模式部署同一声明图，再更新为仅私有模式。存储容器是 ARM 资源；测试 Blob 在下一步上传。工作负载子网明确禁用默认出站访问，虚拟机没有公共 IP。专用终结点使用独立子网和 Blob 私有 DNS 区域。

```
param location string = resourceGroup().location
param prefix string
param adminUsername string = 'azureuser'
@secure()
param sshPublicKey string
@allowed([
```

```
  'Enabled'
  'Disabled'
])
param publicNetworkAccess string = 'Enabled'

var vnetName = '${prefix}-vnet'
var workloadSubnetName = 'workload'
var endpointSubnetName = 'private-endpoints'
var storageName = take(toLower('${prefix}${uniqueString(resourceGroup().id)}'),
24)
var dnsZoneName = 'privatelink.blob.core.windows.net'
var tags = {
  workload: 'network-private-access-lab'
  environment: 'lab'
  owner: 'book-reader'
  costCenter: 'learning'
  expiresOn: 'replace-with-date'
}

resource nsg 'Microsoft.Network/networkSecurityGroups@2024-05-01' = {
  name: '${prefix}-workload-nsg'
  location: location
  tags: tags
  properties: {
    securityRules: [
      {
        name: 'Deny-Internet-Outbound'
        properties: {
          priority: 200
          direction: 'Outbound'
          access: 'Deny'
          protocol: '*'
          sourcePortRange: '*'
          destinationPortRange: '*'
          sourceAddressPrefix: 'VirtualNetwork'
          destinationAddressPrefix: 'Internet'
        }
      }
    ]
  }
}

resource vnet 'Microsoft.Network/virtualNetworks@2024-05-01' = {
  name: vnetName
  location: location
  tags: tags
  properties: {
    addressSpace: { addressPrefixes: [ '10.42.0.0/16' ] }
```

```
    subnets: [
      {
        name: workloadSubnetName
        properties: {
          addressPrefix: '10.42.1.0/24'
          networkSecurityGroup: { id: nsg.id }
          defaultOutboundAccess: false
        }
      }
      {
        name: endpointSubnetName
        properties: {
          addressPrefix: '10.42.2.0/24'
          privateEndpointNetworkPolicies: 'Disabled'
          defaultOutboundAccess: false
        }
      }
    ]
  }
}

resource nic 'Microsoft.Network/networkInterfaces@2024-05-01' = {
  name: '${prefix}-vm-nic'
  location: location
  tags: tags
  properties: {
    ipConfigurations: [
      {
        name: 'ipconfig1'
        properties: {
          privateIPAllocationMethod: 'Dynamic'
          subnet: { id: resourceId('Microsoft.Network/virtualNetworks/subnets',
vnet.name, workloadSubnetName) }
        }
      }
    ]
  }
}

resource vm 'Microsoft.Compute/virtualMachines@2024-07-01' = {
  name: '${prefix}-vm'
  location: location
  tags: tags
  properties: {
    hardwareProfile: { vmSize: 'Standard_B1s' }
    storageProfile: {
      imageReference: {
        publisher: 'Canonical'
```

```
        offer: 'ubuntu-24_04-lts'
        sku: 'server'
        version: 'latest'
      }
      osDisk: { createOption: 'FromImage', deleteOption: 'Delete' }
    }
    osProfile: {
      computerName: '${prefix}-vm'
      adminUsername: adminUsername
      linuxConfiguration: {
        disablePasswordAuthentication: true
        ssh: { publicKeys: [ { path:
'/home/${adminUsername}/.ssh/authorized_keys', keyData: sshPublicKey } ] }
      }
    }
    networkProfile: {
      networkInterfaces: [ { id: nic.id, properties: { deleteOption: 'Delete' }
} ]
    }
    diagnosticsProfile: { bootDiagnostics: { enabled: true } }
  }
}

resource storage 'Microsoft.Storage/storageAccounts@2025-01-01' = {
  name: storageName
  location: location
  tags: tags
  sku: { name: 'Standard_LRS' }
  kind: 'StorageV2'
  properties: {
    allowBlobPublicAccess: false
    allowSharedKeyAccess: true
    minimumTlsVersion: 'TLS1_2'
    publicNetworkAccess: publicNetworkAccess
    networkAcls: {
      bypass: 'None'
      defaultAction: publicNetworkAccess == 'Disabled' ? 'Deny' : 'Allow'
    }
  }
}

resource blobService 'Microsoft.Storage/storageAccounts/blobServices@2023-05-01'
= {
  parent: storage
  name: 'default'
}
resource container
'Microsoft.Storage/storageAccounts/blobServices/containers@2023-05-01' = {
```

```
  parent: blobService
  name: 'lab'
  properties: { publicAccess: 'None' }
}

resource dnsZone 'Microsoft.Network/privateDnsZones@2024-06-01' = {
  name: dnsZoneName
  location: 'global'
  tags: tags
}
resource dnsLink 'Microsoft.Network/privateDnsZones/virtualNetworkLinks@2024-06-
01' = {
  parent: dnsZone
  name: '${prefix}-vnet-link'
  location: 'global'
  properties: {
    registrationEnabled: false
    virtualNetwork: { id: vnet.id }
  }
}

resource endpoint 'Microsoft.Network/privateEndpoints@2024-05-01' = {
  name: '${prefix}-blob-pe'
  location: location
  tags: tags
  properties: {
    subnet: { id: resourceId('Microsoft.Network/virtualNetworks/subnets',
vnet.name, endpointSubnetName) }
    privateLinkServiceConnections: [
      {
        name: '${prefix}-blob-connection'
        properties: {
          privateLinkServiceId: storage.id
          groupIds: [ 'blob' ]
        }
      }
    ]
  }
}
resource dnsZoneGroup
'Microsoft.Network/privateEndpoints/privateDnsZoneGroups@2024-05-01' = {
  parent: endpoint
  name: 'default'
  properties: {
    privateDnsZoneConfigs: [ { name: 'blob', properties: { privateDnsZoneId:
dnsZone.id } } ]
  }
}
```

```
output storageName string = storage.name
output vmId string = vm.id
output storageFqdn string = '${storage.name}.blob.core.windows.net'
output privateEndpointId string = endpoint.id
```

如果你的环境不支持所列某个 API 版本或 defaultOutboundAccess 属性，请选择当前受支持版本，并在实验记录中说明变更。不要悄悄移除私有子网意图；应使用明确等效方案替代，并证明最终出口状态。

7.14.4 引导对象，然后关闭公共路径

```
RG=book-ch07-network
LOCATION=eastus2
PREFIX=ch07demo
SSH_KEY_FILE="$HOME/.ssh/id_ed25519.pub"
mkdir -p evidence

az group create --name "$RG" --location "$LOCATION"   --tags workload=network-
private-access-lab environment=lab owner=book-reader costCenter=learning

az deployment group create   --resource-group "$RG"   --template-file main.bicep
--parameters prefix="$PREFIX" sshPublicKey="$(cat "$SSH_KEY_FILE")"
publicNetworkAccess=Enabled   --output json > evidence/01-bootstrap-
deployment.json

STORAGE=$(az deployment group show --resource-group "$RG" --name main   --query
properties.outputs.storageName.value -o tsv)
VM_ID=$(az deployment group show --resource-group "$RG" --name main   --query
properties.outputs.vmId.value -o tsv)
FQDN="$STORAGE.blob.core.windows.net"

echo 'private-path-ok' > probe.txt
KEY=$(az storage account keys list --resource-group "$RG" --account-name
"$STORAGE"   --query '[0].value' -o tsv)
az storage blob upload --account-name "$STORAGE" --account-key "$KEY"   --
container-name lab --name probe.txt --file probe.txt --overwrite true

az deployment group create   --resource-group "$RG"   --template-file main.bicep
--parameters prefix="$PREFIX" sshPublicKey="$(cat "$SSH_KEY_FILE")"
publicNetworkAccess=Disabled   --output json > evidence/02-private-only-
deployment.json

az storage account show --resource-group "$RG" --name "$STORAGE"   --query
```

```
'{id:id, publicNetworkAccess:publicNetworkAccess,
defaultAction:networkRuleSet.defaultAction}'   --output json > evidence/03-
storage-network-state.json
```

上传发生在禁用公共访问之前，因此读者无需为了植入对象准备第二台私有工作站或专用部署脚本。最终部署状态仅允许私有访问。短暂的引导期属于证据的一部分，不得误认为生产目标状态。

7.14.5 生成短期只读 SAS

```
EXPIRY=$(date -u -v+30M '+%Y-%m-%dT%H:%MZ' 2>/dev/null || date -u -d '+30
minutes' '+%Y-%m-%dT%H:%MZ')
SAS=$(az storage blob generate-sas   --account-name "$STORAGE"   --account-key
"$KEY"   --container-name lab   --name probe.txt   --permissions r   --expiry
"$EXPIRY"   --https-only   --output tsv)
URL="https://$FQDN/lab/probe.txt?$SAS"

# Do not write URL or SAS into the permanent evidence packet.
unset KEY
```

SAS 是一种临时数据平面凭据，只用于让公共和私有网络测试完全相同。它不是推荐的应用身份验证设计。对于支持工作负载身份与最小权限数据角色的服务，生产工作负载应优先采用它们。证据包应记录到期时间和权限集，但不能记录令牌值。

7.14.6 证明公共路径不可用

```
set +e
curl --connect-timeout 10 --max-time 20   --silent --show-error   --dump-header
evidence/04-public-headers.txt   --output evidence/04-public-body.txt   --write-
out '%{http_code}
'   "$URL" | tee evidence/04-public-status.txt
PUBLIC_RC=$?
set -e

echo "curl_exit=$PUBLIC_RC" >> evidence/04-public-status.txt
head -40 evidence/04-public-headers.txt
```

请从 VNet 外部的客户端运行此命令，例如本地开发 Shell 或 Cloud Shell。请求不得返回对象。根据客户端路径和当前 Storage 行为，结果可能是 HTTP 拒绝、连接拒绝或服务特定网络错误。保存状态、标头和正文。验收标准不是某条永久错误字符串，而是禁用公共网络设置后，无法通过公共路径获得对象内容。

7.14.7 证明私有 DNS 和私有数据访问

```
az vm run-command invoke   --ids "$VM_ID"   --command-id RunShellScript   --
scripts "set -e
    echo 'DNS:'
    getent ahostsv4 '$FQDN'
    echo 'CONTENT:'
    curl --fail --silent --show-error '$URL'
  "   --output json > evidence/05-private-run-command.json

jq -r '.value[].message' evidence/05-private-run-command.json
```

DNS 输出必须显示专用终结点子网中的地址，内容输出必须为 private-path-ok。这证明同一 FQDN 已在私有 DNS 上下文中解析，且服务通过 Private Link 接受了连接。SAS 只证明对一个对象、在一个短暂时间窗口内的数据授权；它不会削弱公共路径测试，因为公共客户端使用完全相同的凭据。

7.14.8 检查终结点、DNS、路由和连接证据

```
az network private-endpoint-connection list   --id "$(az storage account show -g
"$RG" -n "$STORAGE" --query id -o tsv)"   --output json > evidence/06-private-
endpoint-connections.json

az network private-dns record-set a list   --resource-group "$RG"   --zone-name
privatelink.blob.core.windows.net   --output json > evidence/07-private-dns-
records.json

az network watcher test-connectivity   --source-resource "$VM_ID"   --dest-
address "$FQDN"   --dest-port 443   --protocol Tcp   --output json >
evidence/08-connection-troubleshoot.json

az monitor activity-log list   --resource-group "$RG"   --offset 2h   --output
json > evidence/09-activity-log.json
```

如果无代理连接故障排除体验或命令参数在当前 CLI 中不同，请使用当前受支持命令并记录调整。证据仍应展示来源、目标 FQDN、端口、可达性结果，以及检测到的所有路由、DNS 或安全问题。可把虚拟网络流日志作为高级扩展启用，但本短实验不要求这样做，因为它们会增加存储、摄取延迟、受支持工作负载约束和成本。

7.14.9 查询网络资源图

```
az extension add --name resource-graph --upgrade

QUERY=$(cat <<'KQL'
```

```
Resources
| where resourceGroup =~ 'book-ch07-network'
| where type in~ (
    'microsoft.network/virtualnetworks',
    'microsoft.network/networksecuritygroups',
    'microsoft.network/networkinterfaces',
    'microsoft.network/privateendpoints',
    'microsoft.network/privatednszones',
    'microsoft.compute/virtualmachines',
    'microsoft.storage/storageaccounts')
| extend publicNetworkAccess=tostring(properties.publicNetworkAccess),
         provisioningState=tostring(properties.provisioningState),
         privateLinkServiceConnections=properties.privateLinkServiceConnections
| project id, type, name, location, publicNetworkAccess, provisioningState,
          privateLinkServiceConnections, tags
| order by type asc, name asc
KQL
)
az graph query --graph-query "$QUERY" --output json > evidence/10-resource-
graph.json
```

资源图应包含 VNet、NSG、NIC、虚拟机、专用终结点、私有 DNS 区域和存储账户。存储账户的 publicNetworkAccess 应报告为 Disabled。Resource Graph 不能替代主动测试；它证明产生观测路径的资源与策略清单。

7.14.10 证据包

工件	最低内容
01-bootstrap-deployment.json	初始部署结果和资源身份。
02-private-only-deployment.json	禁用公共终结点的幂等更新。
03-storage-network-state.json	PublicNetworkAccess 为 Disabled，默认网络操作为 Deny。
04-public-*	公共客户端状态、标头和正文，证明未返回对象。
05-private-run-command.json	私有 DNS 结果和成功获取 private-path-ok 的记录。
06-private-endpoint-connections.json	Blob 子资源已获批准的终结点连接。
07-private-dns-records.json	Blob 私有链接区域中存储账户的 A 记录。

工件	最低内容
08-connection-troubleshoot.json	来源、目标、端口和主动连接结果。
09-activity-log.json	控制平面创建和网络设置变更。
10-resource-graph.json	最终资源清单和公共网络状态。
11-cleanup-proof.json	删除资源组后的零结果查询。
lab-notes.md	区域、API 版本、偏差、成本边界、SAS 到期时间和观测到的错误行为。

7.14.11 常见故障模式

症状	可能原因	要检查的证据
私有虚拟机解析到公共 **IP**	缺少私有 DNS 区域、区域名称错误、无 VNet 链接，或自定义 DNS 缺少转发	getent/nslookup 结果；私有 DNS 区域链接和记录
私有 **DNS** 解析正确，但 **curl** 超时	路由或 NSG 问题；终结点未就绪；虚拟机来宾防火墙；服务路径问题	连接故障排除、有效规则、终结点状态
公共客户端仍可获取对象	公共网络访问未禁用、仍有受信例外，或测试使用了不同终结点	Storage 网络设置、Activity Log、公共请求标头
私有 **curl** 返回授权错误	SAS 已到期、Blob 路径错误、权限不足或时钟偏差	SAS 到期时间和权限；Blob 清单；响应标头
Run Command 在脚本执行前失败	虚拟机代理未就绪、镜像不受支持或平台通信问题	虚拟机实例视图、启动诊断、Run Command 状态
子网属性导致部署失败	API 版本或区域功能不匹配	部署操作详细信息和当前提供商文档
专用终结点连接一直处于待定状态	服务需要手动批准，或连接验证失败	专用终结点连接状态和资源 Activity Log
清理后仍有资源	依赖项删除仍在进行，或某项资源创建在资源组之外	Resource Graph 和按标签进行的订阅范围搜索

7.14.12 清理并证明零残留

```
unset SAS URL
az group delete --name "$RG" --yes --no-wait
az group wait --name "$RG" --deleted

az graph query   --graph-query "Resources | where resourceGroup =~ '$RG' |
project id, type, name"   --output json > evidence/11-cleanup-proof.json
```

最终查询必须不返回任何资源。还应搜索带有工作负载标签的资源，以防诊断资源或区域 Network Watcher 资源创建在实验资源组之外。保留清理证明，并在可行时从 Shell 历史或临时文件中移除短期 SAS。

7.14.13 中国云平台转译

复现验收标准，而不是 Azure 语法。在阿里云上，使用 VPC、vSwitch、安全组、受支持场景中的 PrivateLink 接口终结点或 OSS 网关终结点、PrivateZone 或文档指定的 DNS 集成、服务公共访问设置，以及 VPC 流日志或 ActionTrail 证据。在腾讯云上，使用 VPC、子网、安全组、Private Link 终结点和终结点服务或服务特定私有访问、Private DNS、传送到 CLS 的 Flow Logs，以及 CloudAudit 证据。在火山引擎上，使用 VPC、子网、安全组、PrivateLink、PrivateZone 或受支持的私有解析、VPC 流日志和 CloudTrail。无论使用哪个提供商，都要验证服务支持、终结点类型、审批行为、区域、DNS 自动化、定价，以及真正禁用公共访问的控制。

7.15 CloudDesk 复用：私有访问审计与暴露检测

CloudDesk FinOps Agent 可以复用本章的只读网络推理，而无需获得创建路由、公共 IP、对等互连或终结点的不受限制权限。第一项能力审计托管服务是否具有有效私有路径以及公共路径是否关闭；第二项跨资源、网络、DNS 和服务层清点公共与隐式暴露。变更型修复能力应保持独立，并要求经过批准的计划、所有权验证、成本估算、回滚路径和变更后连接测试。

```
capability: network.privateAccess.audit
risk_level: R0_read_only
inputs:
  service_resource_id: /subscriptions/.../storageAccounts/...
  consumer_scope_id: /subscriptions/.../virtualNetworks/...
outputs:
  private_endpoint_ids
  endpoint_connection_state
  private_dns_zone
  resolved_private_addresses
  public_network_state
  positive_test_evidence
  negative_test_evidence
  findings
  evidence_refs
```

```
capability: cloud.service.publicExposure.detect
risk_level: R0_read_only
inputs:
```

```
  scope_id: /subscriptions/.../resourceGroups/...
outputs:
  public_ip_resources
  internet_facing_frontends
  public_service_endpoints
  implicit_or_explicit_egress_paths
  permissive_network_rules
  stale_dns_or_endpoint_objects
  estimated_network_cost_drivers
  remediation_candidates
  evidence_refs
```

```
capability: network.path.explain
risk_level: R0_read_only
inputs:
  source_resource_id
  destination_fqdn
  protocol: tcp
  port: 443
outputs:
  resolution_chain
  resolved_address
  effective_route
  effective_security_controls
  endpoint_state
  service_network_policy
  authorization_boundary
  likely_failure_domain
  evidence_refs
```

控制平面规则。 未来的 network.privateAccess.enforce 能力是一项高风险变更工具，可能破坏生产访问、备份、监控、构建流水线或受信服务集成。它绝不能与只读审计能力合并。

7.16 关键要点

- 云网络是一组受管理的可达性声明，而不只是拓扑图。
- 名称解析、路由、分段策略、终结点状态、服务前门策略、身份和授权是彼此独立的决策。
- 私有 IP 或专用终结点证明的是路径选择，不会授予数据权限，也不会自动关闭公共终结点。
- 地址规划必须考虑未来的对等互连、传输、Kubernetes 范围、混合网络、灾难恢复和租户边界。
- 没有公共 IP 只关闭一个暴露面。入口、出口、东西向、托管服务、DNS 和管理平面路径必须分别审查。

- NAT 提供出站转换，而不是入站发布或应用感知检查。
- 第 4 层负载均衡、第 7 层路由/WAF、全局边缘交付和集中防火墙检查解决不同问题，并可组合使用。
- 服务终结点、网关终结点和专用终结点不可互换。其地址、路由、策略、DNS、成本和防数据外泄属性各不相同。
- 私有 DNS 是安全和可靠性边界的一部分。使用提供商支持的 FQDN，并从真实客户端上下文证明解析结果。
- 对等互连通常直接且不可传递；大型或混合拓扑需要明确的传输、路由传播和检查设计。
- 配置清单、主动连接测试、流量记录和控制平面审计日志提供互补证据。
- 网络路径具有明确成本和故障域。优化必须保留所需可达性、安全性、性能和恢复结果。
- 多云可移植性始于工程意图和验收标准，而非产品名称匹配。
- 只有私有路径成功、公共路径失败、终结点与 DNS 状态已记录，且清理证明零残留时，实验 6 才算完成。
- CloudDesk 可以复用范围受限的审计和解释能力，而网络变更仍是单独治理的高风险工具。

参考资料与官方文档

[1] 美国国家标准与技术研究院：《NIST 云计算定义》。NIST SP 800-145，2011。https://doi.org/10.6028/NIST.SP.800-145

[2] National Institute of Standards and Technology. Zero Trust Architecture. NIST SP 800-207, 2020. https://doi.org/10.6028/NIST.SP.800-207

[3] IETF. Address Allocation for Private Internets. RFC 1918, 1996. https://www.rfc-editor.org/rfc/rfc1918

[4] IETF. Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan. RFC 4632, 2006. https://www.rfc-editor.org/rfc/rfc4632

[5] IETF. A Border Gateway Protocol 4 (BGP-4). RFC 4271, 2006. https://www.rfc-editor.org/rfc/rfc4271

[6] IETF. Virtual eXtensible Local Area Network (VXLAN). RFC 7348, 2014. https://www.rfc-editor.org/rfc/rfc7348

[7] Microsoft. Azure Virtual Network documentation. Microsoft Learn. https://learn.microsoft.com/en-us/azure/virtual-network/

[8] Microsoft. Secure your Azure Virtual Network deployment. Updated August 13, 2026. https://learn.microsoft.com/en-us/azure/virtual-network/secure-virtual-network

[9] Microsoft. Azure Virtual Network FAQ. Updated July 8, 2026. https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-faq

[10] Microsoft. Azure virtual network traffic routing. Updated March 31, 2026. https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-udr-overview

[11] Microsoft. Azure network security groups overview. Updated July 15, 2025. https://learn.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview

[12] Microsoft. Azure Virtual Network peering. Updated August 12, 2026. https://learn.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview

[13] Microsoft. Default outbound access in Azure. Updated January 30, 2026. https://learn.microsoft.com/en-us/azure/virtual-network/ip-services/default-outbound-access

[14] Microsoft. What is Azure NAT Gateway? Updated July 29, 2026. https://learn.microsoft.com/en-us/azure/nat-gateway/nat-overview

[15] Microsoft. What is Azure Load Balancer? Microsoft Learn. https://learn.microsoft.com/en-us/azure/load-balancer/load-balancer-overview

[16] Microsoft. What is Azure Application Gateway? Updated December 16, 2025. https://learn.microsoft.com/en-us/azure/application-gateway/overview

[17] Microsoft. What is Azure Application Gateway v2? Updated April 10, 2026. https://learn.microsoft.com/en-us/azure/application-gateway/overview-v2

[18] Microsoft. What is Azure Front Door? Updated April 24, 2026. https://learn.microsoft.com/en-us/azure/frontdoor/front-door-overview

[19] Microsoft. Azure Private Link documentation. https://learn.microsoft.com/en-us/azure/private-link/

[20] Microsoft. What is a private endpoint? Microsoft Learn. https://learn.microsoft.com/en-us/azure/private-link/private-endpoint-overview

[21] Microsoft. Azure Private Endpoint private DNS zone values. Updated August 11, 2026. https://learn.microsoft.com/en-us/azure/private-link/private-endpoint-dns

[22] Microsoft. Azure Private Endpoint DNS integration scenarios. Updated September 30, 2025. https://learn.microsoft.com/en-us/azure/private-link/private-endpoint-dns-integration

[23] Microsoft. Azure virtual network service endpoints. Updated July 8, 2026. https://learn.microsoft.com/en-us/azure/virtual-network/virtual-network-service-endpoints-overview

[24] Microsoft. Azure Storage network security overview. Updated July 16, 2025. https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security-overview

[25] Microsoft. Set the default public network access rule for an Azure Storage account. Updated August 26, 2025. https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security-set-default-access

[26] Microsoft. Azure Storage firewall rules and network access control. Updated July 6, 2026. https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security

[27] Microsoft. Azure Storage firewall guidelines and limitations. Updated July 16, 2025. https://learn.microsoft.com/en-us/azure/storage/common/storage-network-security-limitations

[28] Microsoft. Azure Network Watcher overview. Updated February 25, 2026. https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-overview

[29] Microsoft. Connection troubleshoot overview. Updated April 8, 2026. https://learn.microsoft.com/en-us/azure/network-watcher/connection-troubleshoot-overview

[30] Microsoft. Virtual Network Flow Logs overview. Updated June 25, 2026. https://learn.microsoft.com/en-us/azure/network-watcher/vnet-flow-logs-overview

[31] Microsoft. Manage Virtual Network Flow Logs. Updated January 22, 2026. https://learn.microsoft.com/en-us/azure/network-watcher/vnet-flow-logs-manage

[32] Microsoft. Azure VPN Gateway documentation. https://learn.microsoft.com/en-us/azure/vpn-gateway/

[33] Microsoft. Azure ExpressRoute documentation. https://learn.microsoft.com/en-us/azure/expressroute/

[34] Microsoft. Azure Virtual WAN documentation. https://learn.microsoft.com/en-us/azure/virtual-wan/

[35] Microsoft. Azure Firewall documentation. https://learn.microsoft.com/en-us/azure/firewall/

[36] Microsoft. Azure DDoS Protection documentation. https://learn.microsoft.com/en-us/azure/ddos-protection/

[37] Alibaba Cloud. Virtual Private Cloud documentation. https://www.alibabacloud.com/help/en/vpc/

[38] Alibaba Cloud. PrivateLink endpoints - access Alibaba Cloud services privately. Updated May 27, 2026. https://www.alibabacloud.com/help/en/privatelink/create-and-manage-endpoints/

[39] Alibaba Cloud. VPC private access to Alibaba Cloud services. Updated May 27, 2026. https://www.alibabacloud.com/help/en/vpc/vpc-connection-to-cloud-service

[40] Alibaba Cloud. VPC flow logs. Updated June 21, 2026. https://www.alibabacloud.com/help/en/vpc/vpc-flow-logs

[41] Alibaba Cloud. VPC monitoring and logging. Updated April 1, 2026. https://www.alibabacloud.com/help/en/vpc/monitoring-and-logging

[42] Alibaba Cloud. VPC NAT Gateway. Updated May 28, 2026. https://www.alibabacloud.com/help/en/nat-gateway/user-guide/what-is-a-vpc-nat-gateway/

[43] Tencent Cloud. Virtual Private Cloud overview. https://www.tencentcloud.com/document/product/215/535

[44] Tencent Cloud. Tencent Cloud Network overview. https://www.tencentcloud.com/document/product/215/47911

[45] Tencent Cloud. Private Link product overview. https://www.tencentcloud.com/document/product/1216/57917

[46] Tencent Cloud. Cloud Connect Network. https://www.tencentcloud.com/document/product/215/40089

[47] Tencent Cloud. Flow Logs.
https://www.tencentcloud.com/document/product/215/35523

[48] Tencent Cloud. Route tables overview.
https://www.tencentcloud.com/document/product/215/31810

[49] Volcengine. Virtual Private Cloud documentation and VPC overview. Updated June 18, 2026. https://docs.volcengine.com/docs/6396/733270?lang=en

[50] Volcengine. Network FAQ - PrivateLink and VPC connectivity. Updated June 18, 2026. https://docs.volcengine.com/docs/6396/67801?lang=en

[51] Volcengine. NAT Gateway documentation.
https://docs.volcengine.com/docs/6404/106344?lang=en

[52] Volcengine. VPC flow log service documentation.
https://www.volcengine.com/docs/6401/2479863

[53] Volcengine. CloudTrail supported products and control-plane scope. Updated August 28, 2025. https://www.volcengine.com/docs/7015/1142171

第二篇 / 第 8 章

存储、数据湖与事件驱动的数据路径

命名空间、分层、持久性、元数据、事件和证据如何把存储的字节转化为受治理的数据产品

云存储不是廉价的字节容器，而是一组资源契约，决定数据如何命名、访问、保留、复制、转换、观察和退役。数据湖增加共享命名空间、元数据、策略和计算协调。只有通知、幂等性和证据经过一体化设计，事件驱动路径才能把存储数据的变化转化为可靠工作。

本章内容

- 把对象存储、块存储和文件存储视为不同的控制平面与数据平面契约，而非可互换容量。
- 把存储资源建模为命名空间、身份、网络、生命周期、保护、事件、遥测和成本对象构成的图。
- 区分持久性与复制，以及备份、时间点恢复和经过验证的还原程序。
- 依据分类、访问证据、保留义务和经济意图设计数据生命周期策略。
- 把数据湖理解为包含区域、目录、血缘、质量、身份和计算协调的受治理数据资产。
- 构建能够容忍重复通知、延迟交付、乱序和部分故障的事件驱动数据路径。
- 把存储与数据湖决策连接到 AI 原生摄取、检索、评估、可观测性和 FinOps。
- 完成一个独立 Azure 实验，创建、验证并退役对象存储生命周期策略，再把工程意图转译到阿里云、腾讯云和火山引擎。

核心论点。 对象、块、文件和数据湖服务是不同的托管契约。只有命名空间、身份、放置、生命周期、事件语义、证据和经济性经过一体化设计，AI 原生数据路径才会可靠。

现代应用图经常画一个标为“存储”的圆柱体。这个符号很方便，却危险地不完整。数据库卷、共享文件系统、对象语料库、湖仓表、备份仓库和模型工件都可以包含字节，但它们公开的访问单元、延迟特征、命名空间、并发模型、保留语义、故障行为和成本表面并不相同。把它们视作一个类别，会产生易画难运维的架构。

控制平面视角纠正了这种抽象错误。它不只问数据存在哪里，还会问：哪个资源拥有命名空间，谁能更改策略，如何授权数据平面访问，复制覆盖哪些故障，旧数据如何分层或删除，发出哪些事件，什么证据证明转换已经发生，以及如何归属成本。这些问题让存储成为软件工程的一部分，而不是部署后的容量决策。

8.1 存储是一组契约，而非单一服务

首个设计决策不是“选择哪个存储产品？”，而是“工作负载需要何种访问和生命周期契约？”正确答案通常从应用读写的单元开始。

表 8-1 应根据语义而非笼统的数据存在性选择存储契约。

契约	主要单元	典型访问	优势	常见误用
对象存储	由键、类路径名称和元数据寻址的对象	HTTPS/REST、SDK、分段传输	海量规模、低管理成本、分层、事件源、分析语料库	像低延迟 POSIX 磁盘一样挂载，或用于频繁的小规模随机写入
块存储	按规定访问模式附加到一个或多个计算节点的卷	文件系统、数据库引擎、裸设备 I/O	低延迟随机 I/O、可预测性能层、快照	没有集群/文件系统协议却被当作共享数据服务
文件存储	共享层次结构中的文件和目录	SMB、NFS、文件系统 API	迁移兼容性、并发客户端、熟悉的 ACL	在缺少命名空间和元数据规划时被当作无限分析对象存储
数据湖存储	对象/文件加层次命名空间、元数据、治理和分析集成	对象 API、文件系统 API、查询引擎、表格式	大规模分析、共享数据产品、存储与计算解耦	被当作“带文件夹的存储桶”，而目录、血缘、质量和访问仍无人管理



图 8-1 对象、块和文件服务公开不同的资源与数据平面契约。

服务边界还决定提供商管理什么。对于托管磁盘，提供商负责物理介质、所选冗余契约内的复制和卷预配；客户仍负责文件系统、数据库一致性、应用备份和还原验证。对于对象存储，提供商管理分布式对象服务，客户则负责对象命名、身份设计、生命周期筛选器、

元数据质量、事件使用者和恢复语义。更高抽象会减少一些工作，但永远不会消除定义意图的需要。

设计规则。 在写明所需命名空间、访问模式、一致性、持久性、恢复、保留、事件、安全和成本契约之前，不要比较存储产品。

8.2 存储资源图

存储账户或存储桶只是更大资源图的根。其数据平面终结点公开对象、文件、队列或表；控制平面对象定义加密、冗余、网络访问、专用终结点、诊断设置、生命周期规则、不可变保留、复制、事件订阅、角色分配和预算作用域。运维行为来自整张图，而非账户本身。



图 8-2 存储服务周围环绕身份、网络、生命周期、事件、证据和成本资源。

表 8-2 存储设计是一张具有多个控制与证据表面的资源图。

资源关注点	代表性对象	工程问题
身份与授权	角色分配、访问策略、服务主体、托管身份、ACL	哪个主体可以管理资源，哪个主体可以读取或写入数据？
网络边界	防火墙、专用终结点、私有 DNS、服务终结点、路由策略	是否只能通过预期网络路径访问数据平面？
命名空间	账户/存储桶、容器、文件系统、目录、对象键、共享	每个数据产品的持久名称和所有权边界是什么？

资源关注点	代表性对象	工程问题
生命周期与保护	分层规则、保留规则、软删除、版本控制、不可变性、复制	数据何时可以更改、移动、到期、恢复或清除?
事件	事件订阅、主题、队列、筛选器、死信目标	已提交的变更如何成为可靠的下游工作?
证据与经济性	指标、日志、清单、访问记录、成本导出、预算	操作员如何证明状态、归属成本并检测漂移?

资源图必须保留两条独立的权限路径。管理平面权限可以创建账户、改变复制、修改网络策略或安装生命周期规则；数据平面权限可以列出、读取、写入、标记、删除或还原数据。授予其中一项不会自动授予另一项。这种分离对最小权限至关重要：部署身份可以创建存储资源，却不一定有权读取内容；摄取工作进程可以写入对象，却不能禁用不可变性或公共访问限制。

8.3 命名空间与访问语义

命名空间就是架构。它决定应用、策略、事件和治理赖以存在的持久名称。糟糕的命名空间会成为永久耦合机制，因为生命周期筛选器、访问控制、数据产品和下游处理器都会开始依赖它。

8.3.1 扁平对象名称与层次命名空间

对象存储通常公开扁平键空间，即使名称包含斜杠。前缀可以模拟目录，但重命名或删除目录仍可能需要跨许多对象执行操作。层次命名空间把目录变为一等对象，支持原子目录操作和文件系统式授权。Azure Data Lake Storage 为 Blob Storage 增加层次命名空间，让分析引擎能够使用目录语义，同时保留对象规模经济性与分层能力。

层次结构不会自动成为良好数据模型。它应编码来源、数据产品、日期分区、敏感性或处理状态等稳定维度，而不是临时团队名称、用户名或实现细节。稳定分区支持查询剪枝和生命周期策略；不稳定路径会造成昂贵重写和脆弱权限。

表 8-3 命名空间元素应反映稳定的治理和查询维度。

命名空间决策	合适候选	风险候选
顶级所有权	业务数据域或持久产品	当前项目团队或个人用户
处理区域	raw、quarantined、curated、serving	temporary1、final-final、copy
时间分区	具有明确粒度的事件日期	上传时间与业务时间混合

命名空间决策	合适候选	风险候选
租户或安全边界	在端到端设计隔离时使用明确租户标识符	只把租户隐藏在文件名中，而授权共享
架构/版本	版本化数据产品或表元数据	由使用者猜测的隐式格式

8.3.2 ACL、RBAC 与对象级条件

授权可应用于多个层级：云资源作用域、容器或共享、目录层次、对象标签、签名 URL 和应用租户。这些层解决不同问题。云 RBAC 适合在持久资源作用域分配角色；目录 ACL 支持文件系统式委派；对象标签可驱动生命周期筛选器或条件访问；短期签名访问可委派范围狭窄的数据平面操作，而不授予宽泛账户权限。以上任何机制都不能与应用级租户授权混淆。

最小权限边界。 在可使用工作负载身份或短期委派访问时，不要把长期账户密钥放入应用配置。存储密钥通常代表绕过细粒度授权的账户范围权限。

8.4 性能是一项工作负载契约

以 GB 表示的容量并不是完整存储要求。存储性能受对象大小、I/O 大小、并发度、读写比例、随机或顺序访问、元数据操作、位置、网络路径和服务限制影响。系统可能容量充足，却因请求速率、元数据吞吐量、磁盘队列深度或出口带宽饱和而失败。

表 8-4 存储性能必须以工作负载特定术语表达。

维度	对象存储	块存储	文件存储
扩展单元	对象、分区、请求分布	卷性能层和主机附件	共享/文件系统吞吐量和元数据规模
最适合	大型不可变/顺序对象、并行读取器、分析语料库	数据库页、启动/数据卷、低延迟随机 I/O	共享应用文件、用户共享、旧式文件系统预期
常见瓶颈	热点前缀、许多小对象、串行上传、出口	预配 IOPS/吞吐量、虚拟机限制、队列深度	元数据操作、协议延迟、共享吞吐量
主要优化	分区键、分段上传、批处理、并行性	适当调整层级、必要时条带化、缓存、与主机限制对齐	目录设计、客户端并发、缓存、合适协议/层级

大型对象上传通常使用分段或基于块的传输，以便客户端独立重试各部分并利用并行性。这不只是性能技巧，也改变故障恢复、校验和验证与可观测性：操作员应知道是否存在未完成分段上传、哪些部分已提交，以及最终对象是否代表预期内容。

对于分析，将存储与计算解耦可让多个引擎读取同一语料库。这种灵活性使元数据、分区、文件大小、压缩和表格式更加重要。数千个小文件会把廉价对象存储转化为昂贵的元数据与调度问题。存储账单可能仍低，但查询规划和计算成本会上升。

8.5 持久性、可用性、复制、备份与恢复

存储可靠性词汇经常被压缩成“备份”一词。这些术语描述不同机制和承诺。持久性表示已提交数据不丢失的概率；可用性表示服务能否在给定时间满足请求；复制维护额外副本；版本控制保留先前对象状态；软删除推迟不可逆删除；备份创建单独管理的恢复资产或还原点；灾难恢复定义较大故障后如何恢复服务。这些机制有所重叠，但没有任何一个能完全替代其他机制。

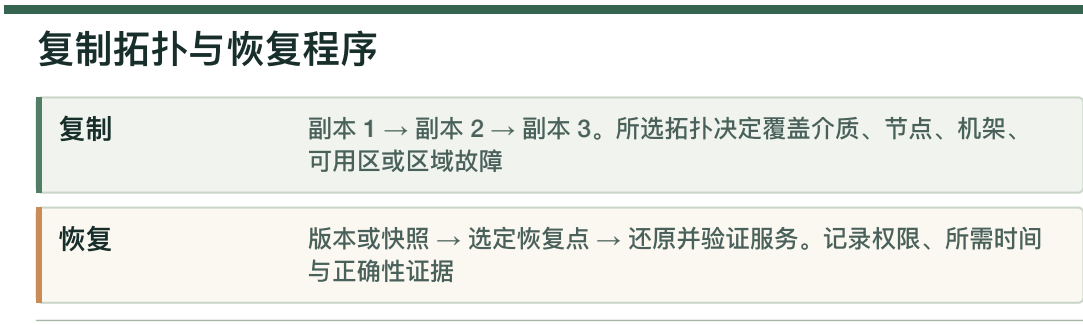


图 8-3 复制保护一种故障拓扑；恢复程序还原有意义的应用状态。

表 8-5 可靠性机制回答不同故障问题。

机制	主要保护	本身不能证明
同步区域复制	所声明拓扑内的介质、节点、机架或可用区故障	从有效的较早逻辑状态恢复
异步跨区域复制	存在非零复制延迟的区域灾难	零数据丢失、应用一致性或即时故障转移
版本控制 / 软删除	在保留窗口内防范意外覆盖或删除	跨多个资源的应用一致恢复
快照	时间点卷或对象状态	快照无损坏、可启动或事务一致
备份与还原程序	定义的恢复点、保留和隔离还原路径	满足 RPO/RTO，除非定期测试
不可变保留	在规定保留期内阻止修改或删除	机密性、原始数据正确性或法律合规本身

有用的可靠性记录会说明受保护故障域、复制模式、预期 RPO、恢复程序、启动恢复所需授权和验证标准。对于数据湖，恢复测试可能不仅要还原对象，还要还原目录元数据、表快照、访问策略、密钥、事件订阅和下游索引。只恢复字节而不恢复元数据，会留下技术上持久却不可用的系统。

可靠性规则。 复制是一种拓扑，恢复是一项运维程序。应把 RPO 和 RTO 视为完整数据服务经过测试的属性，而非从存储 SKU 推断的属性。

8.6 分层、保留、版本控制与不可变性

生命周期管理把数据随时间变化的业务价值转化为存储操作。常见的热—凉—冷—归档递进只是一种策略形态。有些数据保持热态以支持活动事务；有些很快移入归档，因为其价值在于证据而非运维；有些在保留义务到期前不能删除；有些则不应进入归档，因为恢复延迟会违反 SLO。

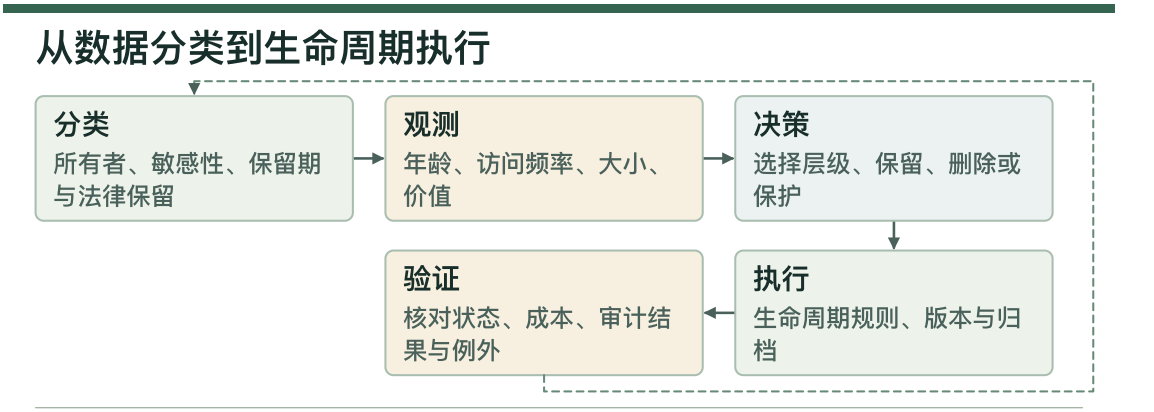


图 8-4 生命周期管理是从分类和观测使用到策略执行与证据的反馈回路。

表 8-6 生命周期功能必须与明确业务和运维语义关联。

生命周期概念	问题	常见故障
访问层	数据需要多频繁、多快地检索？	只优化存储价格，而忽略检索、事务或提前删除成本
保留	数据必须可用或保存多长时间？	只在非正式命名约定中编码法律或业务规则
版本控制	哪些先前对象状态应可恢复？	版本无限增长，没有成本归属或到期规则
软删除	意外删除应在多长时间内可逆？	假设软删除数据已经不再产生存储成本

生命周期概念	问题	常见故障
不可变性 / WORM	必须在哪段时间阻止修改和删除?	锁定策略，却不测试合法更正、法律保全和最终清除
生命周期筛选器	哪些对象位于策略作用域内?	过宽的前缀或标签错误影响无关数据产品
生命周期调度器	提供商会在何时评估并应用规则?	把异步策略当作实时工作流触发器使用

生命周期策略刻意采用异步方式，针对大规模后台评估优化，而不是即时工作流执行。策略可能需要一段时间才能发现合格对象并应用转换。因此，即时验证应检查策略对象、筛选器和当前对象状态；扩展验证则在之后观察提供商应用的转换。若业务流程要求上传后立即行动，应使用事件驱动工作流，而不是生命周期规则。

生命周期与不可变性可能冲突。当不可变保留或法律保全阻止删除时，删除操作无法移除对象。版本控制与软删除也会改变“删除”操作的经济效果，因为已删除内容可能继续保留。治理需要优先级模型：保存控制优先于成本优化控制，例外必须有所有者和到期日期。

8.7 数据湖是受治理的数据资产

数据湖经常被介绍为结构化与非结构化数据的廉价存储。这个描述正确但不充分。生产数据湖是一项受治理资产，其中存储、命名空间、元数据、访问、血缘、质量、保留和计算授权形成共同运维模型。缺少这些控制，数据湖会变成低成本积累的不透明文件，即发现和合规成本很高的数据沼泽。



图 8-5 原始、精选和服务区域需要共享治理平面。

8.7.1 区域表达信任与就绪程度

原始或落地区域保留来源保真度，并提供可恢复的摄取边界。隔离区分离未通过安全、架构或质量检查的文件。精选区域规范架构、解决质量缺陷并建立受治理数据集。服务区域公开针对分析、机器学习、检索或应用使用优化的数据产品。这些区域是责任边界，而不只是文件夹。

表 8-7 数据湖区域编码不同的信任、生命周期和使用者契约。

区域	主要承诺	典型控制	退出条件
原始 / 落地	保留收到的来源和摄取证据	一次写入模式、来源元数据、校验和、受限读取者、保留	对象被接受、隔离或路由以供处理
精选	使数据符合受治理架构和质量契约	验证、血缘、数据分类、去重、分区	数据集通过规定的质量和所有权检查
服务	为使用者工作负载交付稳定数据产品	表/版本契约、性能布局、产品 ACL、SLO、成本归属	使用者可以发现、授权、查询和解释产品
归档	保存低访问历史或证据数据	保留、不可变性、检索程序、法律保全、还原测试	保留期到期，并验证授权清除

8.7.2 目录、血缘、质量与所有权

目录使数据可发现，并把物理位置与架构、分类、所有者和策略关联起来。血缘记录数据如何到达，以及哪些转换生成下游资产。质量规则使就绪程度可衡量。所有权决定谁批准访问、解决缺陷、更改保留并承担成本。这些控制应是可查询的资源或证据，而不是只在电子表格中维护的字段。

数据湖还需要计算治理。查询引擎、Spark 作业、ETL 服务、特征流水线和索引工作进程可能以不同身份和预算读取同一数据。存储解耦并不意味着计算不受约束。平台必须控制哪个引擎能扫描哪个区域、临时结果写到哪里、如何限制出口，以及如何把成本归属给使用工作负载。

8.8 开放表格式把文件转化为托管表

Parquet、ORC 等列式文件可提升分析效率，但一组文件还不是具有事务含义的表。Apache Iceberg、Delta Lake 和 Apache Hudi 等开放表格式增加元数据与提交协议，用于描述快照、架构、分区演进、文件成员关系和并发变更。它们使对象存储可作为托管分析表，而无需每个引擎都从目录列表推断状态。

表 8-8 表格式在对象文件上添加事务元数据层。

能力	为何原始文件不足	表格式增加什么
原子发布	读取者可能看到部分写入的文件集	提交元数据把读取者指向完整表快照
时间旅行 / 回滚	对象版本不一定编码表级一致性	快照历史标识一致的表状态
架构演进	使用者根据文件猜测兼容性	显式架构和字段标识管理变更
分区演进	物理目录布局成为永久查询契约	元数据把逻辑分区与文件路径分离

能力	为何原始文件不足	表格式增加什么
并发写入者	目录列表和重命名约定很脆弱	乐观并发或事务协议协调提交
维护	没有表级视图时，废弃文件不断累积	压缩、保留和孤立文件清理依据元数据执行

开放格式并不意味着无需治理。目录注册、访问策略、加密、保留、快照到期、压缩和成本仍是平台责任。可移植性也分层级：多个引擎可以理解该格式，但目录 API、身份集成、优化功能和运维工具仍可能因提供商而异。多云设计应记录这些不等价之处，而不是宣称完全可移植。

8.9 事件驱动数据路径分离数据平面与事件平面

对象存储是天然事件源，因为对象到达、更新、复制或删除经常启动下游工作，例如缩略图生成、恶意软件扫描、ETL、元数据提取、嵌入生成、索引刷新、通知和合规归档。事件不是数据，而是权威状态发生变化、可能需要处理的通知。



图 8-6 数据平面提交和读取对象；事件平面路由通知并重试工作。

8.9.1 至少一次交付与幂等性

托管事件系统通常采用至少一次交付。订阅者未确认或交付失败时，匹配事件会重试，因此可能出现重复通知；交付还可能延迟或乱序。因此，正确使用者必须幂等：多次处理同一逻辑事件，不应损坏状态或产生重复业务效果。

表 8-9 事件驱动数据路径必须把重试和部分故障视为正常情况。

故障模式	所需设计响应	要保留的证据
重复交付	根据事件 ID、对象版本/ETag 和操作生成幂等键	去重决策和先前完成记录
乱序事件	行动前读取当前对象状态或比较版本/代际	观测版本、预期版本、被拒绝的过期事件

故障模式	所需设计响应	要保留的证据
部分处理	为各阶段设置检查点，并使副作用具备事务性或可补偿	阶段状态、输出标识符、重试次数
有害事件	限制重试并路由到死信存储/队列	原始事件、错误、策略、所有者、重放决策
缺少下游容量	使用队列或持久代理缓冲；施加背压	队列时长、深度、扩缩动作、SLO 影响
事件/数据竞争	重试对象读取，或只响应最终提交事件	读取尝试、最终对象属性、校验和

8.9.2 事件筛选器是契约的一部分

基于事件类型、主题前缀、后缀、对象标签或容器作用域的筛选器可减少噪声并保护下游容量。筛选器应像代码一样审查，因为过宽订阅会对无关对象调用昂贵处理，过窄筛选器则会悄无声息地漏掉所需工作。事件订阅、交付身份、重试策略、死信目标和使用者检查点存储，都是路径的资源级部分。

事件规则。 使用生命周期管理完成基于时间的后台转换；使用事件驱动工作流在数据变化后及时处理。两者解决不同的时间和证据要求。

8.10 AI 原生数据路径

AI 原生应用把数据路径扩展到存储和分析之外。文档会被解析、分类、分块、嵌入、索引、评估、检索、引用，有时还会反馈到模型或智能体行为中。每次转换都会创建新的托管资产，具有自己的所有者、权限、版本、成本和证据。



图 8-7 AI 原生数据路径把源对象连接到受治理知识资产、模型、智能体和证据。

表 8-10 AI 数据路径会创建多个受治理资产，而不是一条“RAG 流水线”。

阶段	托管资产	控制平面问题
落地	源对象加来源元数据	谁可以摄取？对象是否私有、已分类、已校验并按期保留？
解析与规范化	提取的文本、媒体转录、结构化记录	哪个解析器/版本生成了它？如何处理故障和敏感字段？
分块与丰富	块、实体、标签、摘要	什么策略控制块大小、元数据、语言和丰富模型？
嵌入与索引	嵌入向量、搜索文档、表/特征	使用哪个模型/版本、租户、ACL、索引架构、刷新计划和成本？
检索与推理	依据集、提示词上下文、模型/智能体运行	检索前是否应用授权？是否保留引用和跟踪链接？
评估与改进	评估集、故障案例、反馈事件	哪个数据集和版本为晋升、回滚或策略变更提供依据？

存储决策直接影响 AI 质量。来源信息不足会削弱引用；不稳定路径会破坏索引刷新；缺少租户或 ACL 元数据可能导致跨租户检索；无界版本和嵌入会推高成本；归档转换可能让源数据再处理变得异常缓慢。因此，知识系统处于本章存储治理的下游。

原始/精选/服务模式仍然有用。原始区域保存原始文档与摄取证据；精选区域包含经过验证的文本和元数据；服务区域包含搜索索引、向量表示、特征表或其他面向使用者优化的资产。明确区分这些阶段，使再处理、模型迁移、删除和证据收集变得可控。

8.11 安全、隐私与合规

存储系统经常包含组织中范围最广、寿命最长的数据集合。其安全边界必须组合身份、网络限制、加密、分类、保留、审计和恢复。静态加密很重要，但无法弥补公共暴露、过度数据平面角色、共享凭据或不受控导出。

表 8-11 存储安全是一项分层运维契约。

控制	用途	验证问题
工作负载身份	移除长期共享凭据	工作负载能否只获得所需数据平面角色？
私有连接	让数据平面流量保持在批准的网络路径中	公共终结点是否失败，而私有名称解析和访问成功？
加密与密钥策略	保护存储数据并治理密钥权限	谁能使用、轮换、吊销和恢复密钥？

控制	用途	验证问题
分类与 DLP	根据敏感性应用控制	敏感字段是否已标记，并阻止未经授权导出？
不可变性 / 法律保全	在保存期内阻止修改或删除	授权测试能否证明写入/删除遭拒？
访问与活动证据	支持调查与问责	操作员能否关联主体、操作、对象、时间、结果和网络上下文？
删除工作流	遵守保留到期与数据主体或租户删除	决策是否涵盖副本、版本、索引、派生资产和备份？

AI 系统中的删除尤其复杂。移除源对象还可能需要移除解析文本、块、嵌入、搜索文档、缓存上下文、评估示例和派生报告。备份和不可变保留可能延迟最终清除。删除契约必须区分逻辑移除、从活动使用中撤销、保留受保护证据和最终物理清除。

合规边界。 提供商认证和不可变存储功能可以支持控制实现，但本身不能证明组织选择了正确保留期、保护了正确数据，或测试了完整证据与清除工作流。

8.12 存储经济性是多维的

每存储 GB 的最低价格很少等于最低成本架构。存储经济性包括容量、操作、检索、数据传输、复制、元数据、备份副本、提前删除期限、监控、安全扫描，以及处理低效布局所需的计算。成本模型应沿数据路径建立。

表 8-12 存储成本是容量、操作、传输、保护和下游处理之和。

成本表面	典型驱动因素	优化问题
存储容量	当前对象、版本、快照、软删除数据、副本	哪些副本是有意保留、可归属且在保留范围内？
事务	读取、写入、列表、元数据和生命周期操作	批处理、缓存、分区或减少小文件能否降低请求量？
检索	凉/冷/归档读取和再水化	预期访问频率是否适合该层？
数据传输	跨可用区、跨区域、Internet 出口、复制	能否把计算移近数据，或减少结果量？
保护	备份、复制、不可变性、长期保留	哪项恢复或合规目标证明每个副本合理？
分析 / AI 计算	扫描、解析、嵌入、索引、压缩	分区、文件大小、增量更新和缓存是否高效？
运维开销	目录、事件代理、监控、安全服务	控制带来的风险或劳力降低是否足以证明其成本合理？

生命周期策略是一项 FinOps 控制，因为它会改变数据随时间支付的费率。策略应与所有权和价值关联，而非作为通用时间规则应用。例如，90 天后把所有对象移入归档可能降低容量成本，却增加恢复时间、检索费用和运维风险。推荐应包括预期访问模式、预计节省、转换与检索成本、提前删除约束、受影响 SLO，以及回滚或例外流程。

分摊展示应把成本归属到持久数据产品或工作负载，而不只是共享存储账户。标签可用于账户级，但更细分配可能需要对象前缀、元数据、清单导出、访问日志和查询引擎核算。必须记录分配方法，因为近似但稳定的规则，比依据每月变化、看似精确的数字更有用。

8.13 按工程意图进行多云转译

各提供商公开相似的广泛能力系列，但产品名称不能证明运维等价。转译应从工程意图出发：对象语义、层次结构、存储类别、生命周期筛选器、事件交付、数据湖元数据、私有访问、保护和成本证据。下表是导航辅助，而不是声称所有功能或服务限制完全相同。

表 8-13 按工程意图转译存储平台，并在实施前验证不等价细节。

工程意图	Azure	阿里云	腾讯云	火山引擎
对象存储	Blob Storage	Object Storage Service（OSS）	Cloud Object Storage（COS）	Tinder Object Storage（TOS）
层次数据湖	ADLS Gen2 层次命名空间	OSS 加 Data Lake Formation / 分析集成	COS/CHDFS 加 Data Lake Compute 与元数据服务	TOS 加 AI Data Lake 与分析集成
块存储	Managed Disks	Elastic Block Storage / ESSD	Cloud Block Storage（CBS）	Elastic Block Storage（EBS）
文件存储	Azure Files / Azure NetApp Files	NAS / CPFS	Cloud File Storage（CFS）	FileNAS / EFS 系列
生命周期/分层	Blob 生命周期管理；热/凉/冷/归档	OSS 生命周期规则和存储类别	COS 生命周期配置和存储类别	TOS 生命周期规则和存储类型
事件通知	Event Grid 存储事件	OSS 事件通知 / EventBridge 集成	COS 事件通知 / EventBridge / SCF 集成	TOS 事件通知和事件服务
不可变性 / WORM	Immutable Blob Storage	OSS 保留 / WORM 控制	COS 对象锁 / 保留能力	TOS WORM 保留
私有访问	Private Endpoint / Private Link	PrivateLink/VPC 访问模式	PrivateLink/VPC 终结点模式	用于 TOS 和 VPC 访问的 PrivateLink
跨区域复制	对象复制 / 地理冗余选项	OSS 跨区域复制	COS 跨区域复制	TOS 跨区域复制

对于每项实现，至少验证六个不等价点：生命周期应用的单元；最短存储期限和检索行为；事件交付与排序；层次结构和 ACL 模型；复制延迟与故障转移语义；对象级成本与访问证据的可见性。这些细节往往比 API 相似性更重要。

8.14 实验 7：配置并验证对象存储生命周期策略

这个独立实验创建一个 Azure Storage 账户、私有容器、数据保护设置和生命周期管理策略。它在策略前缀内外各上传一个对象，通过控制平面验证规则，手动把一个对象改为凉层以展示即时分层语义，收集证据包，再移除所有资源。

为何包括手动分层。 生命周期评估是异步的，并非按小时内完成的工作流设计。实验立即验证策略意图，再使用直接分层操作证明对象层级行为。可选扩展观察随后可确认提供商应用的生命周期执行。

8.14.1 目标与验收标准

- 把所有实验资源部署到可独立删除的专用资源组。
- 创建标准通用 v2 存储账户，要求安全传输、最低 TLS 1.2、禁用公共 Blob 访问，并且应用工作流不使用共享密钥。
- 分别启用 Blob 版本控制、Blob 软删除和容器软删除等保护控制。
- 创建名为 data 的私有容器。
- 安装一条作用于前缀 data/logs/ 的生命周期规则，使基础 Blob 逐步转到更低访问层并最终删除；先前版本与快照单独到期。
- 在 logs/ 下上传匹配对象，在 reference/ 下上传不匹配对象。
- 验证策略对象、筛选器、操作、资源图和上传对象属性。
- 手动把匹配对象移到凉层，并通过数据平面证明最终层级。
- 删除资源组并证明实验没有资源残留。

8.14.2 前提条件、权限与成本边界

表 8-14 实验前提条件和边界。

项目	要求
工具	Azure CLI、Bicep 和已通过身份验证的 Azure 账户
管理权限	能够创建资源组、部署存储资源并创建范围受限的角色分配
数据权限	实验存储账户上的 Storage Blob Data Contributor（模板将其分配给已登录用户）
区域	支持所选存储和访问层功能的任意 Azure 区域；示例使用 eastus

项目	要求
成本类别	成本较低但不为零：存储字节、读写操作、层级变更和所有保留版本都可能收费
安全边界	使用新资源组；不要把策略指向现有应用或函数主机存储账户
预计时长	即时验证需 20–35 分钟；生命周期调度器观察可选且耗时更长

8.14.3 创建 Bicep 模板

创建名为 main.bicep 的文件。模板使用全局唯一存储账户名称，创建 data 容器，启用版本控制和软删除，安装生命周期规则，并向当前用户授予数据平面访问权限。生命周期前缀相对于存储账户，因此包含容器名称：data/logs/。

```
@description('Azure region for the lab resources.')
param location string = resourceGroup().location

@description('Object ID of the signed-in user who will upload and inspect
blobs.')
param principalId string

var suffix = uniqueString(resourceGroup().id)
var storageName = 'stlake${suffix}'
var blobDataContributorRole = subscriptionResourceId(
  'Microsoft.Authorization/roleDefinitions',
  'ba92f5b4-2d11-453d-a403-e96b0029c9fe'
)

resource storage 'Microsoft.Storage/storageAccounts@2025-01-01' = {
  name: storageName
  location: location
  kind: 'StorageV2'
  sku: { name: 'Standard_LRS' }
  properties: {
    allowBlobPublicAccess: false
    allowSharedKeyAccess: true
    minimumTlsVersion: 'TLS1_2'
    supportsHttpsTrafficOnly: true
    accessTier: 'Hot'
  }
}

resource blobService 'Microsoft.Storage/storageAccounts/blobServices@2025-01-01'
= {
  parent: storage
  name: 'default'
  properties: {
```

```
    isVersioningEnabled: true
    deleteRetentionPolicy: {
      enabled: true
      days: 7
    }
    containerDeleteRetentionPolicy: {
      enabled: true
      days: 7
    }
  }
}

resource dataContainer
'Microsoft.Storage/storageAccounts/blobServices/containers@2025-01-01' = {
  parent: blobService
  name: 'data'
  properties: {
    publicAccess: 'None'
  }
}

resource lifecycle 'Microsoft.Storage/storageAccounts/managementPolicies@2025-
01-01' = {
  parent: storage
  name: 'default'
  properties: {
    policy: {
      rules: [
        {
          name: 'logs-lifecycle'
          enabled: true
          type: 'Lifecycle'
          definition: {
            filters: {
              blobTypes: [ 'blockBlob' ]
              prefixMatch: [ 'data/logs/' ]
            }
            actions: {
              baseBlob: {
                tierToCool: { daysAfterModificationGreaterThan: 30 }
                tierToCold: { daysAfterModificationGreaterThan: 90 }
                tierToArchive: { daysAfterModificationGreaterThan: 180 }
                delete: { daysAfterModificationGreaterThan: 365 }
              }
              version: {
                delete: { daysAfterCreationGreaterThan: 30 }
              }
              snapshot: {
```

```
                delete: { daysAfterCreationGreaterThan: 30 }
              }
            }
          }
        }
      ]
    }
  }
}

resource dataRole 'Microsoft.Authorization/roleAssignments@2022-04-01' = {
  name: guid(storage.id, principalId, blobDataContributorRole)
  scope: storage
  properties: {
    principalId: principalId
    principalType: 'User'
    roleDefinitionId: blobDataContributorRole
  }
}

output storageAccountName string = storage.name
output containerName string = dataContainer.name
output lifecyclePolicyId string = lifecycle.id
```

共享密钥说明。 为广泛兼容存储工具，模板保留共享密钥访问，但实验命令使用 --auth-mode login。确认每个所需客户端都支持 Microsoft Entra 授权后，强化工作负载设计可以禁用共享密钥访问。

8.14.4 部署资源图

```
LAB_RG=book-lab-07-storage-lifecycle
LOCATION=eastus
PRINCIPAL_ID=$(az ad signed-in-user show --query id -o tsv)

az group create --name "$LAB_RG" --location "$LOCATION"

az deployment group create \
  --resource-group "$LAB_RG" \
  --template-file main.bicep \
  --parameters principalId="$PRINCIPAL_ID"

STORAGE=$(az deployment group show \
  --resource-group "$LAB_RG" \
  --name main \
```

```
  --query properties.outputs.storageAccountName.value -o tsv)

echo "$STORAGE"
```

角色分配可能需要短时间传播。如果首个数据平面命令返回授权错误，请稍等并重试，不要退回到账户密钥。

8.14.5 上传匹配与不匹配对象

```
mkdir -p lab-data
printf '2026-08-26T00:00:00Z,INFO,agent-started\n' > lab-data/app.log
printf 'reference material\n' > lab-data/readme.txt

az storage blob upload \
  --account-name "$STORAGE" \
  --container-name data \
  --name logs/app.log \
  --file lab-data/app.log \
  --auth-mode login \
  --overwrite

az storage blob upload \
  --account-name "$STORAGE" \
  --container-name data \
  --name reference/readme.txt \
  --file lab-data/readme.txt \
  --auth-mode login \
  --overwrite
```

第一个对象匹配前缀 data/logs/，第二个对象则位于策略作用域之外。把两个对象放在同一容器中，便于验证筛选边界。

8.14.6 通过控制平面验证策略意图

```
az storage account management-policy show \
  --account-name "$STORAGE" \
  --resource-group "$LAB_RG" \
  --output jsonc

az storage account show \
  --name "$STORAGE" \
  --resource-group "$LAB_RG" \
  --query '{name:name,kind:kind,sku:sku.name,https:enableHttpsTrafficOnly,\
publicBlobAccess:allowBlobPublicAccess,minTls:minimumTlsVersion}' \
  --output table
```

```
az storage blob service-properties show \
  --account-name "$STORAGE" \
  --auth-mode login \
  --output jsonc
```

证据应显示一条名为 logs-lifecycle 的已启用规则、blockBlob 筛选器、前缀 data/logs/、四项基础 Blob 操作，以及分别针对版本和快照的 30 天删除操作。这验证声明的策略，但尚未验证基于时间的转换是否已异步执行。

8.14.7 证明即时分层语义

由于上传对象是新对象，尚不符合生命周期阈值。请使用直接数据平面操作，只把匹配对象移到凉层，再读取其属性。这证明对象支持逐 Blob 分层，且已登录身份可以执行预期操作。

```
az storage blob set-tier \
  --account-name "$STORAGE" \
  --container-name data \
  --name logs/app.log \
  --tier Cool \
  --auth-mode login

az storage blob show \
  --account-name "$STORAGE" \
  --container-name data \
  --name logs/app.log \
  --auth-mode login \
  --query '{name:name,tier:properties.blobTier,etag:properties.etag,\
lastModified:properties.lastModified,versionId:versionId}' \
  --output jsonc

az storage blob show \
  --account-name "$STORAGE" \
  --container-name data \
  --name reference/readme.txt \
  --auth-mode login \
  --query '{name:name,tier:properties.blobTier}' \
  --output jsonc
```

预期结果：logs/app.log 报告 Cool 层，而 reference/readme.txt 保留原始层级。直接操作不能证明调度器已经运行；它证明生命周期策略以后在满足时间条件时将调用的数据平面语义。

8.14.8 构建证据包

创建名为 evidence 的目录并保存以下工件。有用的证据包应允许其他审查者重建意图、观测状态和清理过程，而无需只依赖截图。

表 8-15 建议的生命周期策略证据包。

工件	它证明什么
01-deployment.json	声明的资源部署和输出标识符
02-storage-resource.json	存储种类、SKU、区域、公共访问和传输设置
03-lifecycle-policy.json	规则名称、筛选作用域、阈值和操作
04-blob-service-properties.json	版本控制和软删除保护设置
05-matching-object.json	对象名称、当前层级、ETag、时间和版本标识符
06-nonmatching-object.json	比较对象仍位于策略前缀之外
07-resource-graph.json	实验资源组中创建的资源清单
08-activity-log.json	管理平面创建与策略事件
09-cleanup-proof.json	删除后查询显示没有剩余实验资源

```
mkdir -p evidence

az deployment group show -g "$LAB_RG" -n main -o json \
  > evidence/01-deployment.json
az storage account show -g "$LAB_RG" -n "$STORAGE" -o json \
  > evidence/02-storage-resource.json
az storage account management-policy show -g "$LAB_RG" -n "$STORAGE" -o json \
  > evidence/03-lifecycle-policy.json
az storage blob service-properties show --account-name "$STORAGE" \
  --auth-mode login -o json > evidence/04-blob-service-properties.json
az storage blob show --account-name "$STORAGE" -c data -n logs/app.log \
  --auth-mode login -o json > evidence/05-matching-object.json
az storage blob show --account-name "$STORAGE" -c data -n reference/readme.txt \
  --auth-mode login -o json > evidence/06-nonmatching-object.json

az graph query -q "Resources | where resourceGroup =~ '$LAB_RG' | project
id,type,location,tags" \
  -o json > evidence/07-resource-graph.json

az monitor activity-log list --resource-group "$LAB_RG" --offset 2h -o json \
  > evidence/08-activity-log.json
```

8.14.9 可选的扩展生命周期观察

要观察提供商应用的生命周期执行，请使用非生产账户和其允许阈值适合较长测试的规则。记录规则安装时间、对象修改时间、层级变化时间和所有监控证据。不要假设新规则会立即执行或在可预测分钟执行。生命周期处理是一项周期性、大规模服务，完成时间随作用域和账户对象量变化。

不要只为快速完成演示就在现有生产账户中缩短保留或删除阈值。生命周期策略是一项破坏性控制。请先用狭窄前缀或标签、非关键对象、专用资源组/账户和经过审查的清理计划进行测试。

8.14.10 常见故障及其含义

表 8-16 实验故障揭示生命周期和权限语义，而不只是语法错误。

症状	可能原因	正确响应
上传返回授权失败	角色分配尚未传播或数据平面角色错误	等待，验证分配作用域和主体，再使用登录身份验证重试
策略存在，但对象未立即转换	对象太新，或生命周期调度器尚未处理	验证规则和资格；把生命周期视为异步过程
预期路径之外的对象受到影响	前缀或标签筛选器过宽	禁用规则、检查作用域、必要时还原，再部署经过审查的策略
删除操作看似无效	软删除、版本控制、不可变性或法律保全仍保留数据	检查保护状态，并区分逻辑删除和清除
归档转换失败或不可用	对象/账户类型或区域不支持该操作	检查当前功能支持，并选择兼容层级/操作
清理失败	不可变保留、锁或依赖项仍存在	识别阻塞控制；遵循获授权的退役程序

8.14.11 清理与零残留验证

```
az group delete --name "$LAB_RG" --yes --no-wait
az group wait --name "$LAB_RG" --deleted

az graph query -q "Resources | where resourceGroup =~ '$LAB_RG' | project
id,type" \
  -o json > evidence/09-cleanup-proof.json

cat evidence/09-cleanup-proof.json
```

最终查询应不包含任何资源。还应检查是否有单独作用域的诊断目标、事件订阅、专用终结点、备份保管库、不可变对象或角色分配创建在实验资源组之外。当支持资源作用域更广时，“删除账户”不是充分证据。

8.14.12 把实验转译到中国领先云平台

表 8-17 实验按生命周期和证据意图转译，而非采用完全相同的门户步骤。

实验意图	阿里云	腾讯云	火山引擎
创建对象命名空间	OSS 存储桶与前缀	COS 存储桶与前缀	TOS 存储桶与前缀
定义生命周期	包含存储类别转换和到期的 OSS 生命周期规则	COS 生命周期配置	TOS 生命周期规则
保护先前状态	按需使用版本控制、保留/WORM	按需使用版本控制、对象锁/保留	按需使用版本控制和 WORM/保留
授权数据访问	RAM 角色/策略；临时凭据	CAM 角色/策略；临时凭据	IAM 角色/策略；临时凭据
观察变更	OSS 清单/日志与审计服务	COS 清单/日志与 CloudAudit	TOS 清单/日志与审计服务
触发下游工作	OSS 事件通知 / EventBridge / Function Compute	COS 事件通知 / EventBridge / SCF	TOS 事件通知和函数/事件集成
验证清理	删除存储桶对象/版本和依赖资源，再证明不存在	移除对象/版本、规则、订阅和存储桶	移除对象/版本、生命周期/事件和存储桶

8.15 CloudDesk 复用：面向 FinOps Agent 的存储能力

CloudDesk FinOps Agent 今后可以使用本章中的能力，但本书不实现智能体运行时。云能力契约应保持范围狭窄、可审查并生成证据。

表 8-18 存储章节向 CloudDesk 移交的能力。

能力	模式	用途	所需证据
storage.lifecycle.recommend	只读推荐	识别其访问历史、时间和价值支持更改层级或保留的数据对象或产品	当前层级、时间/访问证据、策略作用域、预计节省、检索/提前删除风险、所有者
storage.cost.optimize	先计划后执行	准备范围受限的生命周期、版本到期或文件布局变更	计划摘要、受影响前缀/标签、成本模型、SLO 影响、审批、回滚/例外

能力	模式	用途	所需证据
storage.event.subscribe	受控变更	为特定数据产品和使用者创建经过筛选的事件订阅	来源、事件类型、筛选器、交付身份、重试/DLQ 策略、订阅者 SLO、审计记录
storage.recovery.verify	读取/测试工作流	评估副本、版本、备份、元数据和还原测试是否满足声明目标	RPO/RTO 目标、最近测试、还原资产 ID、验证结果、差距列表

智能体安全边界。 FinOps 推荐可以识别节省机会，但删除、缩短保留、归档转换、事件订阅或复制变更必须通过所有权、策略、SLO、审批和证据检查。最便宜的数据不一定价值最低或受监管最少。

8.16 关键要点

1. 对象、块、文件和数据湖存储是不同的托管契约；选择首先取决于命名空间和访问语义。
2. 存储资源是一张包含身份、网络、生命周期、保护、事件、遥测、审计和成本的图，而不只是包含字节的终结点。
3. 复制提高对声明故障拓扑的韧性；备份与恢复程序还原有意义的先前状态，并必须依据 RPO 和 RTO 进行测试。
4. 生命周期管理是受治理的反馈回路。筛选器、保留优先级、调度器行为、证据和例外所有权与分层阈值同等重要。
5. 数据湖是受治理的数据资产。区域、目录、血缘、质量、所有权和计算授权可防止低成本存储变成高成本数据沼泽。
6. 开放表格式在对象文件上添加事务元数据，但不会消除目录、身份、维护、保留和成本治理需求。
7. 存储事件采用至少一次交付语义；正确的下游系统会假设存在重复、延迟、乱序和部分故障。
8. AI 原生数据路径会创建多个受治理资产，从源对象到块、嵌入、索引、跟踪和评估证据。存储治理会传递到 AI 质量与隔离。
9. 存储经济性包括容量、操作、检索、传输、保护、元数据和下游计算。每 GB 价格本身不是架构决策。
10. 多云转译应比较工程意图，并记录层次结构、分层、事件交付、复制、保护和证据方面的不等价。

参考资料

1. NIST. The NIST Definition of Cloud Computing, SP 800-145. https://csrc.nist.gov/publications/detail/sp/800-145/final

2. Microsoft Learn. Introduction to Azure Storage.
https://learn.microsoft.com/azure/storage/common/storage-introduction

3. Microsoft Learn. Storage account overview.
https://learn.microsoft.com/azure/storage/common/storage-account-overview

4. Microsoft Learn. Introduction to Azure Blob Storage.
https://learn.microsoft.com/azure/storage/blobs/storage-blobs-introduction

5. Microsoft Learn. Azure Managed Disks overview. https://learn.microsoft.com/azure/virtual-machines/managed-disks-overview

6. Microsoft Learn. Introduction to Azure Files. https://learn.microsoft.com/azure/storage/files/storage-files-introduction

7. Microsoft Learn. Introduction to Azure Data Lake Storage.
https://learn.microsoft.com/azure/storage/blobs/data-lake-storage-introduction

8. Microsoft Learn. Azure Data Lake Storage hierarchical namespace.
https://learn.microsoft.com/azure/storage/blobs/data-lake-storage-namespace

9. Microsoft Learn. Access control model in Azure Data Lake Storage.
https://learn.microsoft.com/azure/storage/blobs/data-lake-storage-access-control-model

10. Microsoft Learn. Azure Blob Storage lifecycle management overview.
https://learn.microsoft.com/azure/storage/blobs/lifecycle-management-overview

11. Microsoft Learn. Configure a lifecycle management policy.
https://learn.microsoft.com/azure/storage/blobs/lifecycle-management-policy-configure

12. Microsoft Learn. Lifecycle management policy structure.
https://learn.microsoft.com/azure/storage/blobs/lifecycle-management-policy-structure

13. Microsoft Learn. Access tiers for blob data. https://learn.microsoft.com/azure/storage/blobs/access-tiers-overview

14. Microsoft Learn. Lifecycle management performance characteristics.
https://learn.microsoft.com/azure/storage/blobs/lifecycle-management-performance-characteristics

15. Microsoft Learn. Data protection overview for Azure Storage.
https://learn.microsoft.com/azure/storage/blobs/data-protection-overview

16. Microsoft Learn. Azure Storage redundancy.
https://learn.microsoft.com/azure/storage/common/storage-redundancy

17. Microsoft Learn. Blob versioning. https://learn.microsoft.com/azure/storage/blobs/versioning-overview

18. Microsoft Learn. Soft delete for blobs. https://learn.microsoft.com/azure/storage/blobs/soft-delete-blob-overview

19. Microsoft Learn. Immutable storage for blob data.
https://learn.microsoft.com/azure/storage/blobs/immutable-storage-overview

20. Microsoft Learn. Point-in-time restore for block blobs.
https://learn.microsoft.com/azure/storage/blobs/point-in-time-restore-overview

21. Microsoft Learn. Reacting to Blob Storage events.
https://learn.microsoft.com/azure/storage/blobs/storage-blob-event-overview

22. Microsoft Learn. Azure Event Grid delivery and retry. https://learn.microsoft.com/azure/event-grid/delivery-and-retry

23. Microsoft Learn. Azure Event Grid event schema for Blob Storage. https://learn.microsoft.com/azure/event-grid/event-schema-blob-storage

24. Microsoft Learn. Azure Event Grid concepts. https://learn.microsoft.com/azure/event-grid/concepts

25. Microsoft Learn. Azure Resource Graph sample queries for Storage. https://learn.microsoft.com/azure/storage/common/resource-graph-samples

26. Microsoft Learn. Microsoft.Storage storageAccounts/managementPolicies Bicep reference. https://learn.microsoft.com/azure/templates/microsoft.storage/storageaccounts/managementpolicies

27. Cloud Native Computing Foundation. CloudEvents Specification. https://cloudevents.io/

28. Apache Iceberg. Documentation and specification. https://iceberg.apache.org/

29. Delta Lake. Documentation. https://docs.delta.io/

30. Apache Hudi. Documentation. https://hudi.apache.org/docs/overview/

31. Apache Parquet. Documentation. https://parquet.apache.org/docs/

32. Alibaba Cloud. Object Storage Service overview. https://www.alibabacloud.com/help/en/oss/product-overview/what-is-oss

33. Alibaba Cloud. OSS storage classes. https://www.alibabacloud.com/help/en/oss/user-guide/overview-55

34. Alibaba Cloud. OSS lifecycle rules. https://www.alibabacloud.com/help/en/oss/user-guide/lifecycle-rules-based-on-the-last-modified-time

35. Alibaba Cloud. OSS event notifications. https://www.alibabacloud.com/help/en/oss/user-guide/event-notifications

36. Alibaba Cloud. Data Lake Formation overview. https://www.alibabacloud.com/help/en/dlf/product-overview/what-is-data-lake-formation

37. Alibaba Cloud. Build a data lake based on OSS. https://www.alibabacloud.com/help/en/oss/use-cases/data-lake

38. Alibaba Cloud. Elastic Block Storage overview. https://www.alibabacloud.com/help/en/ecs/user-guide/block-storage-overview

39. Tencent Cloud. Cloud Object Storage product documentation. https://www.tencentcloud.com/document/product/436

40. Tencent Cloud. COS lifecycle configuration. https://www.tencentcloud.com/document/product/436/17059

41. Tencent Cloud. COS event notification. https://www.tencentcloud.com/document/product/436/16930

42. Tencent Cloud. Data Lake Compute overview. https://www.tencentcloud.com/document/product/1155

43. Tencent Cloud. Cloud Block Storage overview. https://www.tencentcloud.com/document/product/362

44. Tencent Cloud. Cloud File Storage overview. https://www.tencentcloud.com/document/product/582

45. Volcengine. Tinder Object Storage documentation. https://www.volcengine.com/docs/6349

46. Volcengine. TOS lifecycle management. https://www.volcengine.com/docs/6349/119289

47. Volcengine. TOS event notification. https://www.volcengine.com/docs/6349/119286

48. Volcengine. TOS cross-region replication. https://www.volcengine.com/docs/6349/119287

49. Volcengine. TOS PrivateLink access. https://www.volcengine.com/docs/6349/1167162

50. Volcengine. TOS WORM retention. https://www.volcengine.com/docs/6349/147331

51. Volcengine. AI Data Lake documentation. https://www.volcengine.com/docs/6893

52. Volcengine. Elastic Block Storage documentation. https://www.volcengine.com/docs/6397

53. FinOps Foundation. FinOps Framework. https://www.finops.org/framework/

54. OpenTelemetry. Specification. https://opentelemetry.io/docs/specs/

55. OWASP. Cloud-Native Application Security Top 10. https://owasp.org/www-project-cloud-native-application-security-top-10/

应用与云原生控制平面

一次代码提交怎样变成用户实际访问的版本？沿着应用运行时、容器制品、Kubernetes 对象和发布控制器，追踪构建、部署、切流与回滚各自留下的证据。

09 应用 PaaS、函数、事件与 API 管理

10 容器、注册表与软件供应链

11 托管 Kubernetes 与应用定义

12 持续交付、GitOps 与变更治理

第三篇 / 第 9 章

应用 PaaS、函数、事件与 API 管理

托管运行时、事件契约和 API 网关如何把应用代码转化为受治理服务

应用 PaaS 不会消除运维，而是把重复基础设施工作移入托管控制平面契约。Web 应用、函数、事件路由、代理和 API 网关不是可互换的计算选项，而是分别治理运行时、调用、交付、暴露、策略、证据和成本的不同资源。

本章内容

- 把应用托管、函数、事件路由、消息传递、流处理和 API 管理视为不同的托管资源契约。
- 把应用 PaaS 资源建模为计划、应用、工件、配置、身份、终结点、运行状况、扩缩容、遥测和成本对象构成的图。
- 区分构建工件、发布版本、部署槽或修订、运行时实例和外部状态。
- 把函数理解为事件寻址计算，其触发器、托管计划、并发、重试、版本和身份决定运维契约。
- 根据交付语义而非产品名称，在事件通知、代理消息和保留式事件流之间选择。
- 使用 API 管理实现 API 产品化、安全、观察、版本化和治理，同时不把网关与应用中间件或组织策略混淆。
- 把应用 PaaS 决策与身份、私有连接、可观测性、SLO、配额和 FinOps 相连接。
- 完成两个独立 Azure 实验：部署托管 Web API，再把对象创建事件路由到云函数。
- 把工程意图转译到阿里云、腾讯云和火山引擎，并为 CloudDesk FinOps Agent 公开范围受限的能力。

软件团队可以在一下午把 Web 应用从虚拟机迁移到托管应用平台，却仍误解架构。代码运行、提供商修补主机、默认域名出现，但在部署身份、配置、运行状况、扩缩容、发布、网络、证据、成本和退役明确之前，系统并未达到生产就绪。函数与事件也会犯同样错误：短处理程序被当作整个应用，而触发器交付、重试、并发、死信行为和下游权限仍保持隐式。

控制平面视角改变分析单元。它不只问哪个服务能执行代码，还会问哪项托管契约应拥有每项职责。持续运行的 HTTP 服务、事件触发函数、可靠命令队列、高吞吐事件流和 API 网关都会执行或传递应用工作，但承诺不同，故障模式也不同。良好组合是应用架构；忽略这些差异的组合只是产品拼装。

核心论点。 应用 PaaS 是一组托管控制平面契约，以基础设施控制换取生命周期自动化、弹性、策略、遥测和更快交付。应用仍然负责业务行为、数据语义和安全集成。

9.1 PaaS 把重复运维工作产品化

NIST 把平台即服务定义为：使用提供商支持的语言、库、服务和工具，把客户创建或取得的应用部署到提供商支持的基础设施上，而客户不管理底层网络、服务器、操作系统或存储。这一定义仍有用，但现代 PaaS 远不止语言运行时，还可能包含应用托管、函数、事件路由器、消息代理、流式摄取、API 网关、托管配置、身份、诊断、发布原语和策略。

“完全托管”不等于“无需运维”，而是提供商承担重复基础设施职责，并公开更高层运维控制。客户仍要选择发布语义、定义运行状况、配置访问、设计状态、设置配额、处理有害消息、观察业务结果并决定何时退役资源。因此，PaaS 把工作上移：主机级任务减少，应用级契约增加。

表 9-1 PaaS 资源系列把不同形式的应用工作产品化。

资源系列	主要契约	提供商管理的工作	客户决策
托管应用运行时	持续可寻址的服务或 API	操作系统/运行时修补、主机放置、负载分发、平台运行状况	工件、配置、身份、扩缩容、运行状况终结点、发布、数据状态
函数运行时	事件寻址或请求寻址的执行单元	实例创建、事件调度、弹性扩缩、运行时生命周期	触发器、超时、并发、重试、幂等性、版本、下游权限
事件路由器	筛选和路由“某件事已发生”的通知	来源集成、匹配、交付尝试、扇出	事件架构、筛选器、目标、重试期限、死信目标
消息代理	可靠转移命令或工作所有权	持久队列/主题、确认、交付状态、会话或事务	消息契约、竞争使用者、重复处理、死信流程
事件流	大规模保留和重放有序事件序列	分区摄取、保留、使用者偏移量、吞吐基础设施	分区键、保留、重放、使用者组、架构演进
API 管理	使 API 暴露产品化并受治理	托管网关、策略引擎、API 目录、使用遥测	API 契约、使用者身份、配额、路由、版本、产品、后端信任



图 9-1 应用 PaaS 是一组具有共同控制平面义务的托管契约。

这些契约可以组合，但不会相互消除。把应用放在 API 网关之后，并不会把应用运行状况转移给网关；把存储事件连接到函数，并不会让函数成为记录系统；把命令发送到队列，也不能保证业务效果恰好一次。托管平台会减少无差别基础设施工作，但不会消除架构。

9.2 应用 PaaS 是资源图

托管 Web 应用经常被画成标有“App Service”或“SAE”的一个方框。在控制平面中，它是一张图：容量或环境资源提供计算；应用资源绑定运行时与工件；部署槽或修订提供发布隔离；配置与机密引用塑造行为；身份与网络资源治理访问；域名与证书暴露服务；运行状况与扩缩容资源管理运行时状态；遥测和成本资源生成证据。



图 9-2 托管应用是资源图，而非一个不透明托管对象。

表 9-2 应用 PaaS 关注点映射到不同资源和问题。

关注点	代表性对象	工程问题
容量与环境	应用计划、环境、区域、操作系统、实例类别	哪些容量共享、预留或弹性分配，它位于哪个故障域？
应用身份	应用资源、资源 ID、标签、所有者、工作负载身份	持久管理身份是什么，哪个主体可以更改或调用它？
工件与运行时	源包、构建输出、容器镜像、运行时堆栈、启动命令	究竟部署了什么，能否脱离实时运行时重现？
配置与机密	应用设置、环境变量、配置引用、机密存储	哪些值随环境变化，哪些敏感或独立版本化？
发布	槽、修订、部署版本、别名、流量规则	如何验证、晋升、回滚和退役候选版本？
终结点与信任	默认域名、自定义域名、证书、专用终结点、入站策略	谁可通过哪个名称和网络路径访问服务，如何验证使用者身份？
运行状况与弹性	运行状况检查、就绪状态、实例数、自动扩缩规则、重启策略	就绪意味着什么，观测到的需求如何转化为安全容量？
证据与经济性	日志、指标、跟踪、活动记录、配额、成本作用域、预算	操作员如何解释运行时状态、变更历史、性能和成本？

因此，应用资源位于两个契约之间：控制平面创建和配置服务，数据平面接收请求并运行应用代码。编辑应用的管理权限不会自动授予应用级访问权限，应用身份也不会自动获得控制平面权限。这种分离应在设计记录和测试中保持可见。

计划与应用也是不同边界。在 Azure App Service 中，App Service 计划定义一个或多个应用运行所在的区域计算资源。扩缩计划会改变这些应用共享的容量；应用作用域的扩缩或配置则改变该容量契约内的行为。其他提供商使用环境、命名空间或应用级抽象，但问题相同：哪些设置属于共享容量，哪些属于单个应用？

9.3 运行时与部署契约

9.3.1 源代码、包、镜像与运行时

应用 PaaS 可以接受源代码、语言包、编译归档或容器镜像。这些是交付格式，不是等价运行时契约。源代码部署把更多构建行为委托给平台；包部署提供已准备工件，但仍使用提供商运行时；容器部署提供用户空间和依赖项，提供商则继续管理放置、路由、运行状况集成和主机基础设施。

表 9-3 交付格式会改变构建和运行时责任边界。

交付格式	平台构建	客户提供	关键治理问题
源代码	依赖项解析、编译或 buildpack 步骤、运行时镜像选择	源代码、清单、构建设置	能否在实时服务之外重现并证明构建?
编译包	部署提取和运行时托管	构建工件和依赖项	包是否不可变、已校验，并链接到源代码与测试证据?
容器镜像	镜像拉取和托管执行	OCI 镜像、启动行为、暴露端口	镜像是否已签名、已扫描、最小化，并兼容平台契约?
外部构建流水线	仅运行时放置和服务集成	由 CI 生成的版本化工件	是否保留构建身份、来源、审批和工件晋升信息?

构建证据与运行时证据应分开。部署成功只证明工件到达平台，并不能证明它来自批准的源代码修订、通过测试或不存在已知漏洞。容器与供应链章节将详细讨论这些证据。

9.3.2 配置、机密与身份

配置应位于工件之外，使同一工件可穿过各环境。这不意味着所有配置都适合以环境变量公开。功能设置和服务终结点等非敏感值可直接注入；凭据通常应由工作负载身份替代，或从托管机密存储引用。配置变更需要版本和审计语义，因为它们无需改变代码即可改变应用行为。

表 9-4 配置机制具有不同安全与生命周期属性。

机制	适当用途	风险	首选控制
应用设置 / 环境变量	环境特定的非机密行为	未经审查的漂移，或诊断中的意外暴露	IaC 或发布流水线、变更审计、配置清单
设置中嵌入的机密值	旧式连接字符串或密钥	长期凭据泄露且难以轮换	替换为工作负载身份或机密引用
机密存储引用	尚无法消除的密码、证书和 API 密钥	运行时可用性依赖机密存储与身份	托管身份、最小权限、轮换与访问证据
托管服务身份	无需嵌入机密即可访问受支持云服务	角色分配过宽或委派不清	专用身份、狭窄作用域、显式数据平面角色
功能标志 / 动态配置	无需重新部署即可受控改变行为	配置变成不受治理的发布路径	版本、所有者、评估窗口、回滚和遥测

9.3.3 运行状况、就绪、启动与优雅终止

能接受 TCP 连接的进程不一定能正确处理业务请求。应用 PaaS 运行状况必须以应用术语定义。存活信号询问实例是否应重启或替换；就绪询问是否应接收流量；启动信号防止初始化缓慢被过早判定失败；优雅终止为移除实例前完成在途工作留出时间。

表 9-5 运行状况信号回答不同生命周期问题。

信号	问题	典型证据	常见错误
存活	进程是否不可恢复地卡住?	探测成功、心跳、运行时状态	检查每项依赖并引发级联重启
就绪	此实例能否安全接收新流量?	运行状况终结点、依赖就绪、缓存预热状态	在迁移或初始化完成前返回就绪
启动	初始化是否在可接受窗口内完成?	启动探测、初始化跟踪、有界超时	用很长启动时间掩盖损坏依赖项
优雅终止	移除前在途工作能否完成?	排空信号、活动请求数、队列检查点	在确认或提交完成前终止实例
业务运行状况	服务是否交付用户结果?	SLI、合成检查、事务成功率	把 CPU 与进程正常运行时间当作用户体验

只有至少两个健康实例或存在备用路由时，运行状况检查才会产生可靠行为。单实例运行状况探测可以暴露故障，却无法提供冗余。这一区分可防止绿色运行状况指示器被误称为高可用。

9.3.4 扩缩容、容量与状态

托管托管服务可以手动、按计划、按指标或通过平台特定控制器自动扩缩。正如计算章节所述，期望实例数、已分配实例、就绪实例和有效服务容量仍是不同状态。应用 PaaS 隐藏主机预配，但不会隐藏启动延迟、连接池压力、下游配额或有状态瓶颈。

水平扩展假设请求服务状态位于外部、已复制或已安全分区。本地磁盘、内存中会话和单例后台作业会使扩容变成数据丢失或重复工作。持久状态应位于其一致性、可用性、备份和成本契约明确的服务中。后台工作通常应使用队列、调度器或函数触发器，而不是依附在某个 Web 实例上。

9.4 发布：槽、修订、版本与回滚

部署不等于发布。部署把工件放入目标环境；发布改变谁或什么接收流量。托管平台公开多种原语分离这些动作：部署槽、不可变修订、函数版本、别名、流量权重和环境晋升。名称虽不同，工程意图相同。

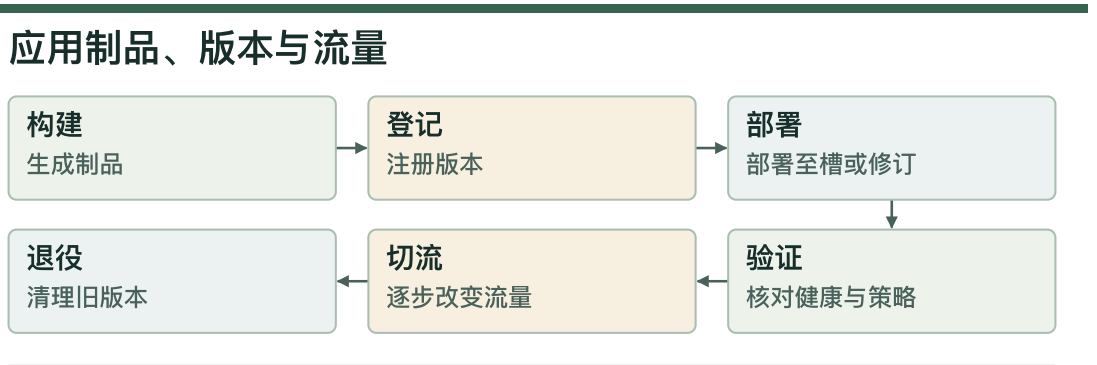


图 9-3 应用生命周期分离工件、发布、运行时、流量和外部状态。

表 9-6 发布原语不可互换。

原语	它隔离什么	它支持什么	它不能解决什么
部署版本	代码与部署配置快照	可追溯性、不可变回滚目标	流量控制或数据回滚本身
槽 / 环境	具有独立终结点或配置的并行运行时	预热验证、交换、蓝绿发布	架构兼容性与状态迁移
修订	托管环境中的不可变应用版本	流量拆分、回滚、审计	未版本化外部配置或共享数据库状态
函数版本	不可变函数代码与选定配置	别名或触发器的稳定目标	事件架构演进与有害消息恢复
别名 / 流量规则	映射到一个或多个版本的稳定逻辑名称	金丝雀、晋升、快速路由回滚	新版本正确性或外部副作用
功能标志	运行时行为开关	较小影响范围与解耦激活	基础设施不兼容或隐藏配置债务

回滚必须说明回滚什么。把流量路由到较早应用版本可能恢复请求处理，却无法撤销数据库迁移、已发出的事件、付款或破坏性工具调用。因此，发布记录应链接工件摘要、配置版本、架构兼容性、迁移计划、审批、运行状况证据和外部影响。

9.5 函数是事件寻址的计算资源

函数常被介绍为一小段代码。在资源层，代码只是一个字段。生产函数还包含托管计划、运行时、内存与超时、并发策略、触发器、绑定或服务客户端、版本、别名、身份、网络路径、重试行为、死信处理、日志、指标、配额和成本作用域。只有忽略这些契约时，函数才显得很小。

函数的触发、执行与输出

数据路径	HTTP、定时器、队列、事件或存储触发 → 函数处理器、运行时与版本 → HTTP 响应、队列、对象或数据库
控制路径	托管计划管理容量与费用；服务身份和 VPC 管理访问；日志、死信队列与指标记录失败

函数的成本和可靠性由计划、并发、超时、身份、网络及重试契约共同决定。

图 9-4 函数是具有触发器、托管、身份、重试和证据契约的事件寻址计算资源。

表 9-7 托管应用和函数针对不同工作负载形态优化。

维度	托管应用运行时	函数运行时
可寻址性	通常持续可用的稳定服务终结点	事件或请求调用函数；实例可能事先不存在
执行生命周期	处理重复请求的长时间运行进程	受平台超时与计划限制的有界调用或任务
扩展单元	应用实例或修订	函数实例与单实例并发
状态假设	可以维护进程本地缓存，但持久状态应位于外部	调用应可重启；持久进度必须外置或设置检查点
最适合	Web/API 服务、持续使用的后端、框架应用	突发事件处理、胶水逻辑、计划任务、扇出工作
主要故障风险	不健康修订、容量滞后、配置漂移、依赖项饱和	重复交付、重试风暴、超时、冷启动、并发冲突
经济性	预配计划或托管容量，通常在分配期间计费	调用、执行时长、内存/CPU、预配或预留容量、网络

9.5.1 触发器与绑定

触发器是函数控制平面契约的一部分。它定义哪个来源可调用函数、事件或消息形态、筛选条件、身份验证、批处理、并发，以及检查点或确认行为。绑定或连接器可以减少样板代码，但不应隐藏权限。操作员仍须知道哪个身份读取来源、哪个身份写入目标，以及失败交付在哪里记录。

表 9-8 常见触发器类型产生不同交付和扩缩容行为。

触发器	调用语义	扩缩信号	所需设计决策
HTTP / Webhook	同步请求—响应	并发请求或平台请求指标	身份验证、超时、幂等性、响应契约

触发器	调用语义	扩缩信号	所需设计决策
计时器 / 计划	平台生成的计划事件	计划与重叠策略	时区、错过的运行、并发执行、回填
队列 / 代理	使用者取得并确认消息	队列长度、时长或分区	可见性锁、重试、有害消息、竞争使用者
事件路由器	通过推送或托管订阅交付通知	事件速率与交付积压	筛选器、重复处理、权威来源查询、死信
对象存储	对象创建或变更通知	对象事件速率或触发器特定轮询器	前缀/后缀筛选器、部分上传对象、重复事件
流 / 日志	由检查点使用的分区记录	分区数、滞后、批大小	排序、检查点、重放、热点分区、架构演进

9.5.2 托管计划与执行经济性

函数托管计划是运维契约，而非定价脚注。它决定缩放到零、预热容量、网络集成、执行时长、实例规格、并发、可用区支持和成本。Azure 当前提供 Flex Consumption、Premium、Dedicated App Service 和旧版 Consumption 选项；正确选择取决于延迟、网络、容量和可靠性要求，而非笼统偏爱“无服务器”。

表 9-9 Azure Functions 托管计划体现不同运维契约。

计划	容量行为	适用场景	设计注意事项
Flex Consumption	采用灵活实例规格和可选始终就绪实例的动态扩缩	需要现代网络和可配置容量的事件驱动工作负载	验证区域支持、触发器行为、冷启动目标和逐函数扩缩
Premium	具有高级网络和长时间运行能力的预热弹性实例	延迟敏感函数、私有网络、持续或突发企业工作	低利用率时，最低分配容量可能主导成本
Dedicated App Service 计划	函数共享已预配 App Service 容量	现有计划容量、可预测始终在线工作负载、组合应用/函数资产	客户负责计划规格与扩缩；隔离不足会产生争用
Consumption	经典动态 Consumption 模型	广泛支持的小型事件驱动工作负载和现有资产	区域、网络、运行时和功能限制可能不同于较新计划

计划名称、限制、运行时版本和区域可用性都是易变事实。实验执行和发布冻结前请验证当前提供商文档。

9.5.3 并发、重试、超时与幂等性

弹性扩缩会放大并发。单次调用正确的处理程序，在十个实例同时处理相关消息时可能失败。因此，并发控制、批处理、下游连接限制、分区键、乐观并发和幂等键都是函数设计的一部分。降低并发可以保护依赖项，但也会增加队列时长和成本。

只有操作可安全重复时，重试才能把暂时故障转化为最终成功。至少一次交付很常见，因此业务操作应使用持久幂等键、事务边界或比较并设置条件。死信目标不是垃圾桶，而是具有所有者、保留、警报、诊断和重放程序的运维工作队列。

9.6 事件、消息与流是不同契约

分布式应用交换事实、命令和历史。云目录经常把事件路由器、队列、主题和流服务放在同一个“消息传递”类别下，但它们的语义不同。最有用的首要问题不是哪个产品最快，而是发布者在断言什么、使用者必须保证什么。



图 9-5 事件、消息和流编码不同意图、所有权、保留与重放契约。

表 9-10 按语义选择事件、消息或流。

契约	发布者声明	使用者义务	代表性能力
事件通知	状态发生了变化	若感兴趣则响应；必要时获取权威状态	筛选、扇出、近实时交付、重试、死信
命令 / 工作消息	执行请求的工作单元	取得所有权、执行、确认、重试或死信处理	队列、主题/订阅、锁、会话、重复检测、事务
事件流	这是有序观测序列	跟踪偏移量、处理分区、容忍重放、维护使用者状态	分区、保留、重放、使用者组、捕获、高吞吐量

Azure Event Grid、Service Bus 和 Event Hubs 清晰体现这种区别。Event Grid 路由离散事件，并与 Azure 资源深度集成；Service Bus 提供企业队列和主题以及会话、重复检测、事务和死信等代理交付功能；Event Hubs 提供分区式、可保留的高吞吐流，支持使用者组和 Kafka 兼容性。阿里云、腾讯云和火山引擎也有相似类别，但确切功能与保证不可互换。

9.6.1 交付语义与故障处理

表 9-11 交付保证需要匹配应用机制。

关注点	平台机制	应用责任
至少一次交付	重试直到成功或到期	幂等性、重复检测、安全重放
排序	分区键、会话、有序队列、单个使用者	选择排序作用域，避免隐藏的跨分区假设
背压	队列深度、使用者滞后、吞吐单元、速率限制	扩缩使用者、卸载、优先处理或减慢生产者
有害工作	交付次数与死信目标	诊断、更正数据或代码、批准重放、保留证据
保留与重放	事件保留、检查点、捕获、归档	版本化使用者和架构；定义重放窗口与副作用安全性
部分失败	逐消息状态、批次重试、事务或补偿	记录进度、隔离故障、避免重试已成功副作用

恰好一次传输不等于恰好一次业务效果。付款、通知、文档转换或资源变更可能跨越具有不同事务边界的系统。稳健设计会记录业务幂等键和结果，再把传输重放视为正常情况。

9.6.2 事件架构、CloudEvents 与 AsyncAPI

事件信封应独立于负载标识事件。CloudEvents 标准化 id、source、type、subject、time、数据内容类型和规范版本等通用元数据，使路由器和可观测性工具无需理解每种业务负载即可运行。负载仍需自己的架构、版本、兼容策略、所有者和保留规则。

```
{
  "specversion": "1.0",
  "id": "01J6-book-lab-0001",
  "source": "/subscriptions/.../storageAccounts/booklab",
  "type": "Microsoft.Storage.BlobCreated",
  "subject": "/blobServices/default/containers/incoming/blobs/sample.txt",
  "time": "2026-08-26T06:00:00Z",
  "datacontenttype": "application/json",
  "data": {
    "url": "https://booklab.blob.core.windows.net/incoming/sample.txt"
  }
}
```

OpenAPI 描述请求—响应型 HTTP 接口；AsyncAPI 描述跨 AMQP、MQTT、Kafka、WebSockets 和 HTTP 等协议的消息驱动接口。成熟平台把这些描述视为受治理工件：经过检查、版本化、发布、链接到实现，并审查兼容性与敏感字段。

9.7 API 管理是产品化控制平面

只有使用者能够发现 HTTP 终结点、获得访问权限、理解契约、在配额内运行，并获得稳定版本与支持，它才成为 API 产品。API 管理为这种产品化提供资源模型和网关运行时。它可作为 App Service 应用、函数、Kubernetes 服务、SaaS 终结点、本地服务或多云后端的前端。



图 9-6 API 管理把 API 产品控制与网关请求执行分离。

表 9-12 API 管理资源治理的不只是 HTTP 转发。

资源	Purpose	生命周期关注点
API 与操作	契约、路径、方法、架构、示例	导入、设计审查、修订、版本、弃用
后端	目标服务与后端身份验证	终结点变更、信任、超时、熔断与凭据轮换
网关	代理并强制实施策略的运行时	容量、放置、网络、升级、遥测
策略	用于安全、验证、转换、路由、速率、缓存或跟踪的运行时规则	作用域、顺序、测试、版本、回滚、性能影响
产品 / 计划	精选 API 包和商业或组织访问契约	受众、配额、条款、所有者、退役
订阅 / 使用者	使用者身份与访问授权	签发、轮换、吊销、使用证据
开发者门户 / 目录	发现、文档、加入和测试	内容发布、身份、暴露、可访问性
工作区 / 团队边界	共享平台中的委派 API 所有权	角色分离、共同策略、命名、成本归属

9.7.1 网关策略与运行时强制实施

网关是 API Management 的数据平面。它验证使用者身份、路由请求、应用策略并记录遥测。策略可以验证令牌和架构、转换标头与正文、重写路由、限制速率、缓存响应、调用辅助服务并发出跟踪。每项策略都会增加运行时复杂性和延迟，因而应有所有者、测试、性能预算、故障行为和回滚路径。

Azure API Management 策略不是 Azure Policy。API Management 策略在网关中针对 API 流量执行；Azure Policy 评估并强制实施 Azure 资源配置；应用中间件是第三层，负责业务授权和领域验证。把三者当作同义词，会产生重复控制或危险缺口。

表 9-13 网关策略、组织策略和应用中间件强制实施不同关注点。

机制	执行点	适合	不适合
API 网关策略	在网关中调用后端之前或之后	使用者身份验证、配额、架构检查、路由、协议转换	深层领域规则、数据库事务、资源级业务授权
云组织策略	控制平面资源创建或评估	允许区域、必需私有访问、SKU 约束、标签、诊断	逐请求授权或负载验证
应用中间件	应用运行时内部	领域授权、租户、事务规则、业务验证	全资产资源治理或跨 API 产品目录
边缘 **WAF / DDoS** 层	Internet 边缘或应用前门	常见 Web 漏洞、流量型防护、边缘筛选	业务配额、API 架构治理、工具级授权

9.7.2 API 生命周期与开发者体验

API 生命周期包括设计、审查、实现、测试、发布、加入、观察、修订、版本控制、弃用和退役。开发者门户不只是文档托管，而是使用者控制平面的一部分，可以公开产品、订阅请求、交互式测试、使用报告、条款、支持联系人和变更通知。

版本与修订解决不同问题。修订在保留公共版本契约的同时改变 API 实现或策略；版本引入有意的契约破坏或长期变体。平台应定义旧版本保留多久、哪些使用者依赖它们，以及退役前需要何种证据。

9.7.3 API 管理不是负载均衡器或服务网格

表 9-14 相邻流量组件功能有所重叠，但主要控制平面不同。

组成部分	主要受众	主要抽象	典型作用域
第 **4** 层负载均衡器	网络和平台操作员	IP、端口、后端池、运行状况探测	传输层分发
第 **7** 层应用网关 / 入口	应用平台操作员	主机、路径、证书、路由、WAF	Web 入口与应用路由
API 管理	API 生产者和使用者	API、操作、产品、订阅、策略、开发者门户	API 生命周期与使用者治理
服务网格	服务平台和应用团队	服务身份、东西向流量、策略、遥测	内部服务到服务通信

组成部分	主要受众	主要抽象	典型作用域
应用中间件	应用开发者	领域请求、用户、租户、事务	业务行为与授权

9.8 身份、网络与信任边界

托管应用路径包含多个身份。最终用户可能向身份提供程序验证；网关验证使用者身份并应用产品策略；应用接收委派用户令牌或网关断言；应用工作负载身份访问存储、数据库、队列或模型资源；事件路由器或代理使用交付身份调用函数或终结点。每一跃都有独立受众、凭据、作用域和审计轨迹。



图 9-7 托管应用路径是一条身份与证据链，而非一个受信网络。

表 9-15 身份链应在每一跃保持明确。

跃点	主体	授权决策	证据
使用者到网关	用户、应用、合作伙伴、设备、智能体	此使用者能否调用该 API 产品和操作?	令牌声明、订阅、策略结果、网关请求日志
网关到后端	网关身份、证书、托管身份、签名请求	网关能否调用该后端终结点?	后端身份验证结果、路由和策略跟踪
应用到托管服务	应用或函数工作负载身份	此工作负载能否读写目标数据平面资源?	角色分配、服务访问日志、跟踪跨度
事件源到路由	服务身份或可信发布者	该来源能否向总线或主题发布此事件类型?	事件源配置、发布记录
路由或代理到函数	交付服务或托管触发器身份	该订阅能否调用函数并交付此项?	订阅、触发器配置、交付尝试、函数调用
操作员到控制平面	人员或自动化身份	此主体能否更改应用、函数、路由、API 或策略资源?	角色分配、活动日志、部署记录

网络控制仍然必要，但不能替代身份。App Service VNet 集成主要控制应用进入虚拟网络的出站访问；专用终结点控制应用的入站私有访问。其他提供商也有相似区分。API 网关和函数也可能具有公共、VPC 集成或私有终结点。正如网络章节所述，DNS、路由、防火墙和服务访问策略必须一致。

9.9 可观测性、SLO、配额与成本

应用 PaaS 产生多种运维数据源。平台指标揭示实例数、CPU、内存、请求率、调用次数、队列时长、事件交付失败、网关延迟和配额使用；应用遥测揭示业务结果与依赖行为；活动日志揭示谁改变了资源；成本记录揭示预配容量和使用量。连贯的服务视图按资源 ID、部署版本、跟踪 ID、事件 ID、使用者和成本作用域关联这些数据。

表 9-16 每项 PaaS 资源都需要自己的运维证据。

资源	主要 SLI 或信号	重要故障证据	成本驱动因素
托管应用	成功请求率、延迟、业务事务成功率	不健康实例、部署失败、启动失败、依赖项饱和	计划或实例容量、执行时间、网络、日志
函数	成功调用率、时长、队列时长或触发器滞后	超时、重试次数、有害项、限制、冷启动、依赖故障	调用次数、内存/CPU 时长、预配容量、网络
事件路由器	交付成功率、交付延迟、死信数量	筛选器不匹配、交付到期、终结点不可访问	操作或事件、交付、数据传输
消息代理	队列时长、活动消息、完成率、死信数量	锁丢失、重复、有害消息、会话积压	代理容量、操作、存储、吞吐量
事件流	入口速率、使用者滞后、分区偏斜、捕获成功率	限制、热点分区、检查点失败、保留缺口	吞吐或处理单元、入口、保留、捕获
API 网关	网关成功率、后端延迟、策略延迟、配额拒绝	身份验证失败、后端超时、策略错误、容量饱和	网关实例/层级、请求或数据处理、网络、遥测

SLO 应沿用户可见路径定义。网关可能健康，但后端失败；函数可能执行成功，但处理事件太晚；队列可能保留每条消息，但业务积压违反目标。应定义“99% 的已接受上传在五分钟内生成处理结果”等复合指标，再把路径拆解为事件交付、函数执行、下游写入和最终可用性。

单个请求可能产生多种成本：网关处理、应用容量、函数执行、消息操作、存储事务、遥测摄取和出站数据。最便宜的组件不一定形成最便宜架构。无服务器执行会减少闲置计算，但可能增加请求、日志和数据传输费用；持续分配的应用可能更适合稳定流量。

表 9-17 PaaS 成本包络应包含每个托管层。

层	典型计费依据	优化问题
应用托管	实例或计划规格与分配时间，有时基于请求自动扩缩	应用能否安全共享容量、缩容或移到更合适计划？
函数	调用次数、内存/CPU 时长、预配或始终就绪实例	工作能否批处理、缩短时长、控制并发或避免不必要触发？
事件与消息	操作、吞吐容量、保留、存储、交付	筛选器是否减少噪声、批大小是否适当、保留是否匹配重放需求？
API 管理	网关容量/层级、请求、数据处理、网络	哪些 API 需要高级隔离、缓存、转换或全局放置？
可观测性	日志与跟踪摄取、保留、查询、导出	采样、维度、保留和诊断类别是否与决策关联？
网络与安全	专用终结点、NAT、负载均衡、WAF、出口	安全边界是否证明路径合理，并消除意外公共传输？

9.10 Azure 参考控制平面

Azure 提供连贯参照，因为它在 Azure Resource Manager 下把托管应用、函数、事件路由、代理消息、流和 API 管理公开为独立资源。资源彼此集成，但对象模型仍清晰可见。

表 9-18 Azure 参考资源及其控制平面用途。

能力	关键资源	控制平面用途
应用托管	App Service 计划、Web App、部署槽、应用设置、托管身份、专用终结点、诊断设置	定义运行时容量、应用生命周期、发布隔离、身份、访问、运行状况、遥测和成本作用域
函数	Function App、托管计划、函数、版本、触发器、绑定或连接、托管身份	定义事件寻址执行、容量、并发、网络、重试、版本和证据
事件路由	Event Grid 系统主题或自定义主题、事件订阅、筛选器、交付终结点、死信目标	通过重试和交付证据把可信来源的事件匹配到目标
代理消息	Service Bus 命名空间、队列、主题、订阅、规则、死信队列	通过确认、会话、事务、重复与死信处理传递命令或工作
流处理	Event Hubs 命名空间、事件中心、分区、使用者组、捕获目标	为独立使用者和重放摄取并保留分区流
API 管理	API Management 服务、网关、API、操作、后端、策略、产品、订阅、门户、工作区	实现 API 产品化、强制运行时策略、吸纳使用者、版本化契约并收集使用证据

App Service 自动扩缩与 Azure Monitor 自动扩缩彼此相关但属于不同机制。自动扩缩允许平台为受支持的 Web 应用和计划作出扩容决策；Azure Monitor 自动扩缩针对受支持资源使用明确指标或计划规则。应记录选择，因为它改变阈值、响应时序和证据的所有者。

Azure Functions 托管选择同样影响契约。Microsoft 目前建议在可用区域中为许多动态扩缩应用使用 Flex Consumption；Premium 和 Dedicated 计划仍适合预热容量、高级网络、可预测执行或共享 App Service 资源。现有 Consumption 工作负载仍然有效，但应根据当前区域和功能要求评估。

不应按熟悉程度选择 Event Grid、Service Bus 与 Event Hubs。Event Grid 适合离散事件路由；Service Bus 适合可靠的企业命令和工作；Event Hubs 适合高容量保留式流。API Management 则跨 Azure、本地或其他云托管的后端提供托管 API 生命周期和网关数据平面。

9.11 多云转译

各提供商用不同产品边界公开相近工程意图。转译必须保留不等价性。例如，提供商可能把应用托管与函数执行组合在一个产品系列中，或通过专注 Kubernetes 和微服务的云原生网关公开 API 管理。正确映射识别的是资源契约，而不只是最接近的产品名称。

表 9-19 按工程意图进行多云转译。

工程意图	Azure	Alibaba Cloud	Tencent Cloud	Volcengine
托管应用运行时	App Service; selected Container Apps scenarios	Serverless App Engine (SAE)	Tencent Cloud Elastic Microservice (TEM)	veFaaS 应用和 Web 应用形式；与 App Service 仅部分等价
函数运行时	Azure Functions	Function Compute 3.0	Serverless Cloud Function (SCF)	Function as a Service (veFaaS)
事件路由	Event Grid system/custom topics and subscriptions	EventBridge system/custom event buses, rules, targets	EventBridge buses, rules, connectors and targets	veFaaS 触发器加服务特定事件中心与集成；不假设存在完全等价的通用事件总线
可靠消息传递	Service Bus queues and topics	Simple Message Queue, RocketMQ, RabbitMQ and related services	TDMQ family and queues/topics	RocketMQ, Kafka, BMQ and related managed messaging
流处理	Event Hubs with Kafka compatibility	Message Queue for Apache Kafka and streaming services	CKafka and stream services	Kafka and cloud-native message engine BMQ

工程意图	Azure	Alibaba Cloud	Tencent Cloud	Volcengine
API 管理	Azure API Management	传统 API Gateway 和云原生 API Gateway，包括 AI 网关能力	Tencent Cloud API Gateway	API Gateway (APIG), including VKE discovery and AI gateway functions
身份与证据	Entra ID/RBAC, managed identity, Azure Monitor and activity logs	RAM, service roles, CloudMonitor, SLS and ActionTrail	CAM, service roles, Cloud Monitor, CLS and CloudAudit	IAM, service roles, Cloud Monitor, TLS and audit services

提供商产品边界与国际站点可用性会变化。请把表格视为意图映射，再验证当前区域、版本、API 版本、网络选项、限制和计费模型。

表 9-20 选择工作表可防止“无服务器”成为唯一要求。

问题	托管应用	函数	事件路由器 / 代理 / 流	API 管理
什么启动工作?	HTTP、RPC、后台循环、调度器	请求或事件触发器	发布者、生产者或源服务	API 使用者请求
工作运行多久?	持续进程或重复请求	有界调用或任务	独立于使用者运行时的交付与保留	逐请求网关执行
谁拥有工作完成责任?	应用实例及其服务契约	函数调用与下游事务	使用者确认、检查点或订阅状态	后端应用；网关只拥有策略与路由
什么状态是持久的?	外部数据库、存储、队列或托管会话存储	外部状态、检查点、幂等性记录	消息/事件保留、偏移量、死信记录	API 定义、策略、使用者权限、使用证据
主要故障模式是什么?	不健康修订、容量、依赖项或配置	重复、超时、重试风暴、冷启动、并发	积压、到期、有害项、分区偏斜、重放缺口	身份验证/策略错误、后端超时、容量或契约破坏

9.12 实验 8：部署托管 Web API

本实验验证一项能力：托管应用资源可以接受不可变应用工件、公开稳定 HTTPS 终结点、报告应用定义的运行状况、携带托管身份、发出日志，并作为一张作用域明确的资源图删除。实验不构建完整 CI/CD 流水线或 API 网关；它们是独立控制平面。

9.12.1 目标、架构与成本边界

表 9-21 实验 8 契约。

项目	决策
能力目标	在 Azure App Service 上部署并验证最小 ASP.NET Core Web API。

项目	决策
主要资源	资源组、Linux App Service 计划、Web App、系统分配的托管身份、应用设置和日志配置。
证据	部署记录、资源 JSON、HTTPS 响应、/health 响应、运行状况检查配置、身份主体 ID、应用日志、活动日志、清理证明。
成本边界	Basic 计划按分配时间收费。使用一个小型实例，迅速完成实验并删除资源组。
未演示	多可用区可用性、流量拆分、自定义域名、专用终结点、数据库状态或生产 SLO。

前提条件：Azure 订阅、Azure CLI、.NET SDK、zip 工具，并有权创建资源组、App Service 计划、Web 应用、身份和诊断设置。命令假设使用 Bash，且所在区域提供 Linux App Service。

```
export SUFFIX=$(openssl rand -hex 3)
export LOCATION=eastus
export RG=book-ch09-app-$SUFFIX
export PLAN=plan-ch09-$SUFFIX
export APP=bookapi-$SUFFIX

az login
az account show --output table
az group create --name "$RG" --location "$LOCATION"
```

9.12.2 创建最小 API

```
dotnet new web -n ManagedBookApi --framework net8.0
cd ManagedBookApi
cat > Program.cs <<'CS'
var builder = WebApplication.CreateBuilder(args);
builder.Logging.ClearProviders();
builder.Logging.AddConsole();

var app = builder.Build();

app.MapGet("/", () => Results.Ok(new {
    service = "managed-book-api",
    version = "1.0.0",
    lab = Environment.GetEnvironmentVariable("BOOK_LAB_ID")
}));

app.MapGet("/health", () => Results.Ok(new {
    status = "healthy",
```

```
    utc = DateTimeOffset.UtcNow
}));

app.MapGet("/api/info", (HttpContext context, ILogger<Program> log) => {
    var correlation = context.TraceIdentifier;
    log.LogInformation("BOOK_API request {CorrelationId}", correlation);
    return Results.Ok(new {
        correlationId = correlation,
        machine = Environment.MachineName,
        framework =
System.Runtime.InteropServices.RuntimeInformation.FrameworkDescription
    });
});

app.Run();
CS

dotnet publish -c Release -o publish
(cd publish && zip -r ../app.zip .)
```

应用公开一个与业务无关的根响应、应用定义的运行状况终结点，以及记录关联标识符的终结点。工件不包含环境特定的终结点或凭据。

9.12.3 创建托管运行时

```
az appservice plan create   --resource-group "$RG"   --name "$PLAN"   --location
"$LOCATION"   --is-linux   --sku B1

az webapp create   --resource-group "$RG"   --plan "$PLAN"   --name "$APP"   --
runtime "DOTNETCORE:8.0"

az webapp identity assign   --resource-group "$RG"   --name "$APP"

az webapp config appsettings set   --resource-group "$RG"   --name "$APP"   --
settings BOOK_LAB_ID=lab-8 SCM_DO_BUILD_DURING_DEPLOYMENT=false

az webapp config set   --resource-group "$RG"   --name "$APP"   --always-on true
--http20-enabled true   --generic-configurations '{"healthCheckPath":"/health"}'

az webapp log config   --resource-group "$RG"   --name "$APP"   --application-
logging filesystem   --level information   --web-server-logging filesystem
```

如果 Azure CLI 版本不接受通过 --generic-configurations 设置 healthCheckPath，请在门户中把 Health check 路径设为 /health，或通过当前 App Service REST API 更新站点配置。在证据包中记录所用方法。

9.12.4 部署工件并验证数据平面

```
az webapp deploy   --resource-group "$RG"   --name "$APP"   --src-path app.zip
--type zip

export BASE="https://$APP.azurewebsites.net"
curl -fsS "$BASE/"
curl -fsS "$BASE/health"
curl -fsS "$BASE/api/info"
```

三个请求都必须返回 HTTP 成功。保存响应正文。运行状况响应只证明此实例当前能够处理运行状况路径，并不能证明多实例或可用区韧性。

9.12.5 验证控制平面与证据

```
az webapp show -g "$RG" -n "$APP"   --query
'{id:id,state:state,host:defaultHostName,httpsOnly:httpsOnly,identity:identity.p
rincipalId}'   --output json > webapp-resource.json

az webapp config show -g "$RG" -n "$APP"   --query
'{linuxFxVersion:linuxFxVersion,alwaysOn:alwaysOn,http20Enabled:http20Enabled,he
althCheckPath:healthCheckPath}'   --output json > webapp-config.json

az monitor activity-log list   --resource-group "$RG"   --offset 2h   --query
'[].
{time:eventTimestamp,operation:operationName.value,status:status.value,caller:ca
ller}'   --output json > activity-log.json

az webapp log download -g "$RG" -n "$APP" --log-file appservice-logs.zip
```

检查日志归档，或短暂使用 az webapp log tail，然后再次调用 /api/info 并找到 BOOK_API 记录。如果所选配置不支持文件系统日志下载，请捕获平台当前日志流证据并记录限制。证据包应包含 app.zip 哈希、部署输出、资源与配置 JSON、HTTPS 响应、日志证据、活动日志和简短责任声明。

9.12.6 清理与零残留证明

```
cd ..
az group delete --name "$RG" --yes --no-wait
az group wait --name "$RG" --deleted
az resource list --resource-group "$RG" --output table 2>/dev/null || true
```

最终资源查询应不返回任何资源。还要确认没有自定义 DNS、证书、外部监控工作区或部署工件创建在资源组之外。记录删除时间戳。

9.12.7 常见故障模式与转译

表 9-22 实验 8 故障模式。

症状	可能原因	证据与修正
默认站点加载成功，但 **/health** 失败	工件或启动命令未正确部署	检查部署输出、应用日志、运行时堆栈和发布文件夹布局
已配置运行状况路径，但没有可用性收益	只有一个实例	说明限制；只有在独立可靠性练习中才扩展到两个或更多实例
身份存在，但服务访问失败	未向托管身份分配数据平面角色	检查主体 ID 与目标角色分配；不要把插入机密作为首个修复
日志为空	日志类别、提供程序、保留或时序错误	调用 /api/info，检查日志配置，等待摄取，再查询所选接收端
出现意外持续成本	删除应用后 App Service 计划仍处于分配状态	删除整个资源组并验证计划已消失

在阿里云上，使用 SAE 应用复现意图：部署源包或镜像，配置运行状况与扩缩容，分配 RAM/服务角色，启用 ARMS 或 SLS 证据，并删除应用与命名空间资源。在腾讯云上，使用 TEM、CAM、日志、运行状况和应用访问。在火山引擎上，根据需要使用 veFaaS 应用或 Web 应用形式，但要记录它与持续分配的 App Service 计划之间的不等价。

9.13 实验 9：把对象事件路由到云函数

本实验独立于实验 8 验证第二项能力：Azure Storage 可以发布对象创建事件，由 Event Grid 筛选并路由到具名 Azure Function，再通过平台证据建立关联。事件按至少一次交付处理；函数记录事件身份，而不假设唯一性。

9.13.1 目标、架构与成本边界

表 9-23 实验 9 契约。

项目	决策
能力目标	创建 Event Grid 触发函数，并证明上传一个 Blob 会产生相关联的函数调用。
主要资源	资源组、存储账户和 incoming 容器、Function App 与托管计划、Application Insights 资源、Event Grid 系统主题/订阅。
证据	函数与触发器清单、事件订阅 JSON、上传的 Blob 属性、Event Grid 交付证据、包含事件 ID 与主题的函数跟踪、活动日志、清理证明。
成本边界	使用面向 Consumption 的函数计划和小型存储账户。日志和事件操作可能产生少量费用；请迅速删除资源组。

项目	决策
交付假设	至少一次。重复交付有效，不得产生重复业务效果。

前提条件：Azure CLI、Azure Functions Core Tools、Python 3.11、zip，以及创建存储、Function App、Event Grid、监控和角色分配资源的权限。示例使用经典 Consumption 计划，命令简洁且广为熟悉。生产环境应根据当前网络、延迟、并发和区域要求评估 Flex Consumption、Premium 或 Dedicated。

```
export SUFFIX=$(openssl rand -hex 3)
export LOCATION=eastus
export RG=book-ch09-event-$SUFFIX
export STORAGE=bookevt$SUFFIX
export FUNC=book-event-fn-$SUFFIX
export SUB=blob-created-to-function

az group create --name "$RG" --location "$LOCATION"
az storage account create   --name "$STORAGE"   --resource-group "$RG"   --
location "$LOCATION"   --sku Standard_LRS   --kind StorageV2

az functionapp create   --resource-group "$RG"   --name "$FUNC"   --storage-
account "$STORAGE"   --consumption-plan-location "$LOCATION"   --runtime python
--runtime-version 3.11   --functions-version 4   --os-type Linux
```

9.13.2 创建并发布事件函数

```
mkdir FunctionEventLab && cd FunctionEventLab
func init . --python -m V2

cat > function_app.py <<'FUNCTION_PY'
import json
import logging
import azure.functions as func

app = func.FunctionApp()

@app.function_name(name="OnBlobCreated")
@app.event_grid_trigger(arg_name="event")
def on_blob_created(event: func.EventGridEvent):
    data = event.get_json()
    record = {
        "event_id": event.id,
        "event_type": event.event_type,
        "subject": event.subject,
        "event_time": str(event.event_time),
```

```
        "url": data.get("url") if isinstance(data, dict) else None
    }
    logging.info("BOOK_EVENT %s", json.dumps(record, sort_keys=True))
FUNCTION_PY

cat > requirements.txt <<'REQUIREMENTS'
azure-functions
REQUIREMENTS

func azure functionapp publish "$FUNC" --python
```

处理程序记录 Event Grid 事件 ID、类型、主题、事件时间和对象 URL。日志有意保持无副作用，因此重放安全。生产处理程序应使用 event.id 或业务幂等键保护所有持久写入或外部操作。

9.13.3 创建容器与事件路由

```
export SUBSCRIPTION_ID=$(az account show --query id -o tsv)
export USER_OBJECT_ID=$(az ad signed-in-user show --query id -o tsv)
export STORAGE_ID=$(az storage account show -g "$RG" -n "$STORAGE" --query id -o
tsv)
export
FUNCTION_ID="/subscriptions/$SUBSCRIPTION_ID/resourceGroups/$RG/providers/Micros
oft.Web/sites/$FUNC/functions/OnBlobCreated"

az role assignment create   --assignee-object-id "$USER_OBJECT_ID"   --assignee-
principal-type User   --role "Storage Blob Data Contributor"   --scope
"$STORAGE_ID"

az storage container create   --account-name "$STORAGE"   --name incoming   --
auth-mode login

az eventgrid event-subscription create   --name "$SUB"   --source-resource-id
"$STORAGE_ID"   --endpoint-type azurefunction   --endpoint "$FUNCTION_ID"   --
included-event-types Microsoft.Storage.BlobCreated   --subject-begins-with
"/blobServices/default/containers/incoming/"
```

事件订阅作用于存储账户，筛选 BlobCreated 事件，并缩小到 incoming 容器。Azure Function 资源 ID 是目标。Event Grid 管理 Azure Function 目标类型的终结点验证。保存订阅对象，并记录平台创建的任何系统主题。

Event Grid CLI 和嵌套 Azure Function 资源 API 是易变界面。若提供商 CLI 变化，请保留资源意图：源存储账户、BlobCreated 筛选器、incoming 容器主题前缀、Azure Function 终结点类型和具名函数资源。

9.13.4 上传对象并关联调用

```
printf 'application-paas-event-lab
' > sample.txt

az storage blob upload   --account-name "$STORAGE"   --container-name incoming
--name sample.txt   --file sample.txt   --auth-mode login   --overwrite

az storage blob show   --account-name "$STORAGE"   --container-name incoming   -
-name sample.txt   --auth-mode login   --output json > uploaded-blob.json

az eventgrid event-subscription show   --name "$SUB"   --source-resource-id
"$STORAGE_ID"   --output json > event-subscription.json
```

等待部署、触发器同步、事件交付和遥测摄取完成。找到属于 Function App 的 Application Insights 资源，再查询 BOOK_EVENT 记录。

```
export AI=$(az resource list   --resource-group "$RG"   --resource-type
microsoft.insights/components   --query '[0].name' -o tsv)

echo "Application Insights resource: $AI"

az monitor app-insights query   --resource-group "$RG"   --app "$AI"   --
analytics-query "traces | where message startswith 'BOOK_EVENT' | project
timestamp, message | order by timestamp desc | take 20"   --output json >
function-traces.json

cat function-traces.json
```

跟踪必须包含以 /incoming/sample.txt 结尾的主题和非空 Event Grid 事件 ID。如果 CLI 无法在当前环境查询 Application Insights，请使用 Application Insights Logs 体验或受支持日志流，保存查询和结果并记录限制。当机器可读输出可用时，不要只用截图替代证据。

9.13.5 构建证据包

```
mkdir -p evidence

cp event-subscription.json evidence/01-event-subscription.json
cp uploaded-blob.json evidence/02-uploaded-blob.json
cp function-traces.json evidence/03-function-traces.json

az functionapp show -g "$RG" -n "$FUNC" -o json   > evidence/04-function-
app.json
```

```
az resource show --ids "$FUNCTION_ID" --api-version 2022-03-01 -o json   >
evidence/05-function-resource.json

az monitor activity-log list   --resource-group "$RG"   --offset 2h   --query
'[].
{time:eventTimestamp,operation:operationName.value,status:status.value,caller:ca
ller,resourceId:resourceId}'   --output json > evidence/06-activity-log.json

sha256sum function_app.py requirements.txt sample.txt   > evidence/07-artifact-
hashes.txt
```

表 9-24 实验 9 证据包。

工件	它证明什么	它不能证明什么
事件订阅 JSON	来源、目标类型、筛选器、主题前缀、预配状态	所有未来事件都会在 SLO 内处理
上传的 Blob 属性	对象身份、容器、大小、ETag 和修改时间	事件已交付
包含事件 ID 的函数跟踪	具名函数接收并编入索引的一个 Event Grid 信封	恰好一次交付或持久业务完成
Function App 和函数资源 JSON	运行时资源、函数身份、区域和部署状态	此处理程序未演练正确下游授权
活动日志	谁创建或更改了控制平面资源	数据平面上传和内部重试详细信息
工件哈希	使用了哪些源代码与测试输入	源代码审查质量或供应链来源本身

撰写一段解释，把上传 Blob 的 ETag 和时间戳与 Event Grid 主题及 BOOK_EVENT 跟踪关联起来。明确说明 Event Grid 使用可重试交付，重复事件有效。未来有副作用的处理程序必须在确认完成前持久化去重或幂等记录。

9.13.6 故障模式及其含义

表 9-25 实验 9 的故障揭示事件契约语义。

症状	可能原因	证据与修正
函数未显示为 Azure Function 目标	函数部署或触发器索引尚未完成	检查 Function App 部署日志与函数清单；等待触发器同步
Blob 上传成功，但没有跟踪	订阅筛选器、终结点、遥测摄取或运行时启动错误	检查事件订阅预配状态、主题前缀、Function App 日志和 Event Grid 指标

症状	可能原因	证据与修正
事件订阅创建遭拒	终结点资源 ID、来源 ID、角色、区域或架构契约无效	将资源 ID 与当前 CLI 文档比较；使用 Azure Function 终结点类型，而非通用 Webhook
跟踪出现多次	预期的至少一次重试，或第二次上传/覆盖生成了另一个事件	比较事件 ID；使副作用幂等，而不是抑制有效交付
用户无法创建容器或上传 **Blob**	缺少数据平面角色分配，或分配尚未传播	验证存储账户作用域的 Storage Blob Data Contributor；等待并重试，不要使用账户密钥
找不到 **Application Insights** 资源	Function App 遥测已禁用或以不同方式创建	启用批准的遥测接收端、重复事件并保存查询结果和配置证据

9.13.7 清理与零残留证明

```
cd ..
az group delete --name "$RG" --yes --no-wait
az group wait --name "$RG" --deleted

az graph query   -q "Resources | where resourceGroup =~ '$RG' | project id,type"
--output json > cleanup-proof.json

cat cleanup-proof.json
```

最终查询应不包含任何资源。确认 Event Grid 系统主题、事件订阅、Application Insights 资源、存储账户、Function App 和角色分配随资源组删除。若遥测工作区、死信目标或自定义主题创建在组外，应单独移除并包含相关证明。

9.13.8 把实验转译到中国领先云平台

表 9-26 按事件源、路由、函数、证据和清理意图转译实验 9。

意图	Alibaba Cloud	Tencent Cloud	Volcengine
对象事件源	OSS 事件或 EventBridge 来源	COS 事件通知或 EventBridge 来源	TOS 触发器或受支持服务事件源
路由资源	EventBridge 事件总线与规则，或 OSS 到 Function Compute 集成	EventBridge 总线与规则，或 COS 到 SCF 集成	veFaaS 触发器配置和受支持事件集成
计算目标	Function Compute 函数和版本/别名	SCF 函数和版本/别名	veFaaS 函数或应用
交付证据	EventBridge/FC 指标、日志、规则和目标配置	EventBridge/SCF 日志、规则和交付证据	veFaaS 触发器配置、调用日志和服务指标

意图	Alibaba Cloud	Tencent Cloud	Volcengine
故障契约	所选路由支持的重试、死信与函数错误处理	受支持的重试、死信与 SCF 错误处理	触发器特定的重试与日志语义；记录服务差异
清理证明	删除规则、函数和 OSS 资源，并确认没有 RAM/日志资源残留	删除规则、SCF 和 COS 资源，并确认没有 CAM/日志资源残留	删除触发器、函数/应用、TOS 资源和外部日志接收端

各平台不公开完全相同的资源图。转译保留工程意图：具名对象事件源、可筛选路由、版本化函数目标、显式交付行为、机器可读证据和经过验证的退役。

9.14 CloudDesk 复用：范围受限的应用能力

CloudDesk FinOps Agent 今后可以调用本章介绍的托管能力，但不得获得通用部署 Shell、不受限制的 API 网关管理员角色或任意函数调用凭据。每项能力应公开狭窄架构、作用域身份、风险类别、适当情况下的计划或试运行、证据信封，以及清理或回滚语义。

表 9-27 应用 PaaS 章节向 CloudDesk 移交的能力。

能力	Purpose	风险	所需证据
app.hosting.deploy	把批准的应用工件部署到具名托管运行时，并返回终结点、版本、身份和运行状况状态	R3——受控变更	工件摘要、目标资源 ID、配置差异、部署结果、运行状况结果、活动日志、成本作用域
app.health.check	读取应用运行状况配置，并执行范围受限的运行状况或合成检查	R0/R1——观察	资源 ID、运行状况路径、响应、实例状态、时间戳、部署版本
function.invoke	以经过验证的输入和有界超时调用一个允许列表函数	R1-R3，取决于副作用	函数版本、调用者身份、请求/事件 ID、结果、时长、下游证据和成本
event.route.create	在审查计划后创建或更新范围狭窄的事件路由	R3——受控变更	来源、筛选器、目标、重试/死信设置、计划摘要、审批、预配状态、交付测试
api.gateway.publish (optional extension)	把经过审查的 API 契约和范围受限的网关策略发布到批准环境	R3——受控变更	OpenAPI 摘要、后端、策略差异、安全审查、部署结果、合成测试和回滚目标

前四项能力是本章架构包移交内容。可选 API 能力之所以列出，是因为 API 管理属于本章资源模型；只有定义了使用者身份、契约审查、策略作用域和回滚之后才应启用。

9.15 关键要点

- 应用 PaaS 不是单一产品类别。托管应用运行时、函数、事件路由器、代理、流和 API 网关是不同的控制平面契约。
- 托管应用是一张资源图，包含计划、应用、工件、配置、身份、终结点、发布、运行状况、扩缩容、遥测、策略、配额、成本和退役对象。
- 构建工件、部署版本、槽或修订、运行时实例、流量映射、配置和外部状态必须分别版本化和治理。
- 函数是事件寻址计算。触发器语义、托管计划、并发、重试、超时、版本、身份和证据都是函数资源的一部分。
- 事件宣布状态变化，消息分配工作，流保留有序观测。产品名称不会抹去这些语义差异。
- 至少一次交付使重复处理和幂等业务效果成为正常要求，而非例外防御。
- API 管理通过契约、版本、产品、订阅、策略、开发者体验、网关运行时和证据实现 API 产品化。
- 网络可达性、网关策略、工作负载身份、事件交付权限和后端授权是独立信任决策。
- SLO 必须沿网关、应用、事件、函数、代理和下游状态构成的用户可见路径定义；仅组件健康并不充分。
- PaaS 经济性包括预配容量、执行、事件与消息操作、API 网关处理、遥测、私有连接和数据传输。
- 多云转译应保留资源意图、责任边界、交付语义、证据和成本，同时记录不等价之处。
- CloudDesk 应使用狭窄且生成证据的应用能力，而非不受限制的云管理接口。

参考资料

1. NIST. The NIST Definition of Cloud Computing, SP 800-145. https://csrc.nist.gov/publications/detail/sp/800-145/final

2. Microsoft Learn. Azure App Service overview. https://learn.microsoft.com/azure/app-service/overview

3. Microsoft Learn. App Service plans. https://learn.microsoft.com/azure/app-service/overview-hosting-plans

4. Microsoft Learn. Set up staging environments in Azure App Service. https://learn.microsoft.com/azure/app-service/deploy-staging-slots

5. Microsoft Learn. Monitor App Service instances by using Health check. https://learn.microsoft.com/azure/app-service/monitor-instances-health-check

6. Microsoft Learn. Scale up and scale out an Azure App Service app.
https://learn.microsoft.com/azure/app-service/manage-scale-up

7. Microsoft Learn. Automatic scaling in Azure App Service.
https://learn.microsoft.com/azure/app-service/manage-automatic-scaling

8. Microsoft Learn. Managed identities for App Service and Azure Functions.
https://learn.microsoft.com/azure/app-service/overview-managed-identity

9. Microsoft Learn. Integrate an app with an Azure virtual network.
https://learn.microsoft.com/azure/app-service/overview-vnet-integration

10. Microsoft Learn. Use private endpoints for Azure Web App.
https://learn.microsoft.com/azure/app-service/overview-private-endpoint

11. Microsoft Learn. Configure diagnostic logging for apps in Azure App Service.
https://learn.microsoft.com/azure/app-service/troubleshoot-diagnostic-logs

12. Microsoft Learn. Azure Functions overview.
https://learn.microsoft.com/azure/azure-functions/functions-overview

13. Microsoft Learn. Azure Functions triggers and bindings concepts.
https://learn.microsoft.com/azure/azure-functions/functions-triggers-bindings

14. Microsoft Learn. Azure Functions hosting options.
https://learn.microsoft.com/azure/azure-functions/functions-scale

15. Microsoft Learn. Azure Functions concurrency.
https://learn.microsoft.com/azure/azure-functions/functions-concurrency

16. Microsoft Learn. Azure Functions error handling and retries.
https://learn.microsoft.com/azure/azure-functions/functions-bindings-error-pages

17. Microsoft Learn. Azure Event Grid trigger for Azure Functions.
https://learn.microsoft.com/azure/azure-functions/functions-bindings-event-grid-trigger

18. Microsoft Learn. Work with Event Grid triggers and bindings in Azure Functions.
https://learn.microsoft.com/azure/azure-functions/event-grid-how-tos

19. Microsoft Learn. Introduction to Azure Event Grid.
https://learn.microsoft.com/azure/event-grid/overview

20. Microsoft Learn. Azure Event Grid concepts.
https://learn.microsoft.com/azure/event-grid/concepts

21. Microsoft Learn. System topics in Azure Event Grid.
https://learn.microsoft.com/azure/event-grid/system-topics

22. Microsoft Learn. Azure Event Grid delivery and retry.
https://learn.microsoft.com/azure/event-grid/delivery-and-retry

23. Microsoft Learn. Azure Service Bus messaging overview.
https://learn.microsoft.com/azure/service-bus-messaging/service-bus-messaging-overview

24. Microsoft Learn. Azure Event Hubs overview.
https://learn.microsoft.com/azure/event-hubs/event-hubs-about

25. Microsoft Learn. Compare Azure messaging services.
https://learn.microsoft.com/azure/service-bus-messaging/compare-messaging-services

26. Microsoft Learn. Azure API Management key concepts.
https://learn.microsoft.com/azure/api-management/api-management-key-concepts

27. Microsoft Learn. Azure API Management policies.
https://learn.microsoft.com/azure/api-management/api-management-howto-policies

28. Microsoft Learn. Import and publish an OpenAPI specification in Azure API Management. https://learn.microsoft.com/azure/api-management/import-and-publish

29. Microsoft Learn. Azure API Management workspaces.
https://learn.microsoft.com/azure/api-management/workspaces-overview

30. Microsoft Learn. Azure messaging services documentation.
https://learn.microsoft.com/azure/messaging-services/

31. Cloud Native Computing Foundation. CloudEvents Specification.
https://cloudevents.io/

32. OpenAPI Initiative. OpenAPI Specification.
https://spec.openapis.org/oas/latest.html

33. AsyncAPI Initiative. AsyncAPI Specification.
https://www.asyncapi.com/docs/reference/specification/latest

34. OpenTelemetry. Specification. https://opentelemetry.io/docs/specs/

35. FinOps Foundation. FinOps Framework. https://www.finops.org/framework/

36. Alibaba Cloud. Serverless App Engine documentation.
https://www.alibabacloud.com/help/en/sae/

37. Alibaba Cloud. Function Compute documentation.
https://www.alibabacloud.com/help/en/functioncompute/

38. Alibaba Cloud. EventBridge documentation.
https://www.alibabacloud.com/help/en/eventbridge/

39. Alibaba Cloud. Message Queue for Apache RocketMQ documentation.
https://www.alibabacloud.com/help/en/apsaramq-for-rocketmq/

40. Alibaba Cloud. Cloud-native API Gateway documentation.
https://www.alibabacloud.com/help/en/api-gateway/

41. Alibaba Cloud. Resource Orchestration Service resource types for API and application services. https://www.alibabacloud.com/help/en/ros/developer-reference/list-of-resource-types-by-service

42. Tencent Cloud. Tencent Cloud Elastic Microservice documentation.
https://www.tencentcloud.com/document/product/1094

43. Tencent Cloud. Serverless Cloud Function documentation.
https://www.tencentcloud.com/document/product/583

44. Tencent Cloud. EventBridge documentation.
https://www.tencentcloud.com/document/product/1108

45. Tencent Cloud. TDMQ documentation.
https://www.tencentcloud.com/document/product/1179

46. Tencent Cloud. API Gateway documentation.
https://www.tencentcloud.com/document/product/628

47. Volcengine. Volcano Engine Function as a Service documentation.
https://www.volcengine.com/docs/6662

48. Volcengine. veFaaS overview. https://www.volcengine.com/docs/6662/162366

49. Volcengine. TOS-triggered function service example.
https://www.volcengine.com/docs/6662/1119948

50. Volcengine. API Gateway documentation. https://www.volcengine.com/docs/6569

51. OWASP. API Security Top 10. https://owasp.org/API-Security/

52. IETF. Idempotency-Key HTTP Header Field. https://datatracker.ietf.org/doc/draft-ietf-httpapi-idempotency-key-header/

53. Microsoft Azure Well-Architected Framework. Reliability design principles. https://learn.microsoft.com/azure/well-architected/reliability/principles

54. Microsoft Azure Architecture Center. Asynchronous messaging options. https://learn.microsoft.com/azure/architecture/guide/technology-choices/messaging

55. Microsoft Learn. Azure Retail Prices API overview. https://learn.microsoft.com/rest/api/cost-management/retail-prices/azure-retail-prices

第三篇 / 第 10 章

容器、注册表与软件供应链

内容寻址工件、注册表控制平面和可验证证据如何把镜像转化为受治理发布单元

容器化改变的不只是打包方式。它把运维发布边界从已配置机器移到由摘要标识、通过注册表分发的 OCI 工件，并用有关组成、来源、完整性、风险、审批和预期用途的证据环绕它。最终的软件供应链是一套云控制平面，而不是一串 Docker 命令。

本章内容

- 把容器镜像视为内容寻址发布单元，而不是可移植虚拟机。
- 使用 OCI 镜像、索引、描述符、分发、主体和引用项契约推理镜像及派生证据。
- 区分可变标签、不可变摘要、语义发布版本、部署修订和晋升记录。
- 把构建建模为一项围绕源代码、基础镜像、依赖项、参数、机密、构建器身份、网络访问和目标平台的受治理契约。
- 设计最小化、非 root、多阶段镜像，同时区分体积小与真正的安全性、可维护性。
- 把注册表同时作为云控制平面和工件数据平面运维，并分别履行身份、网络、保留、复制和证据义务。
- 理解 SBOM、来源、签名、扫描结果、VEX 文档和测试证明能够与不能证明什么。
- 在源代码审查、构建、注册表、晋升、准入和运行时响应各阶段实施策略门禁。
- 把供应链推理扩展到模型权重、分词器、适配器、提示词、评估集和安全策略等 AI 原生工件。
- 完成一个独立 Azure 实验，按摘要构建、推送、扫描、签名和验证镜像，再生成可复用证据包。
- 把工程意图转译到阿里云 ACR、腾讯云 TCR 和火山引擎 CR，并为 CloudDesk FinOps Agent 公开范围受限的能力。

最简单的容器工作流有四条命令：构建、标记、推送和运行。它足以演示可移植性，却不足以建立信任。生产使用者需要知道哪个源代码修订生成了这些字节、哪些基础镜像和依赖进入构建、哪个构建器执行它、结果是否经过扫描、谁授权发布、工件是否被修改，以及同一工件是否到达目标环境。只有把镜像与证据视为受治理资源，注册表才能回答这些问题。

本章所依据的课程顺序刻意注重实践：构建多阶段镜像、扫描、推送到私有注册表、授权拉取身份、附加签名并验证结果。本章保留该顺序，但提升分析单元。主题不再是孤立的 Dockerfile 或 ACR 仓库，而是从源代码到运行时的完整发布路径，每个边界都有身份和可验证证据。

核心论点。 当容器镜像的不可变摘要连接到可信构建证据、获授权注册表操作、策略决策和运行时验证时，它才成为受治理发布单元。没有来源信息的可移植性，只是可移植的不确定性。

10.1 容器化移动发布边界

传统服务器交付经常在一台可变机器上组合多项职责：操作系统配置、运行时安装、应用二进制文件、服务账户、证书和运维脚本。部署可能只因这台特定机器积累了预期状态而成功。重现相同环境需要机器镜像，或能够重建它的配置过程。

容器改变了该边界。运行时实例仍是受命名空间、cgroup、文件系统挂载、能力、seccomp 和其他操作系统控制约束的进程，但可部署应用文件系统由镜像组装。镜像可在部署前构建、独立于主机标识、在兼容注册表间复制，并由不同运行时实例化。主机与编排器仍然重要，但不再以隐式本地状态包含应用发布。

这种转变正是容器改变 PaaS 的原因。早期应用平台通常围绕受支持语言和框架版本组织。容器镜像把交付单元移向显式运行时工件。托管应用平台、无服务器容器服务、Kubernetes、批处理系统和 AI 服务平台都能使用同一镜像，同时为网络、扩缩容、身份、运行状况和成本应用不同控制平面契约。

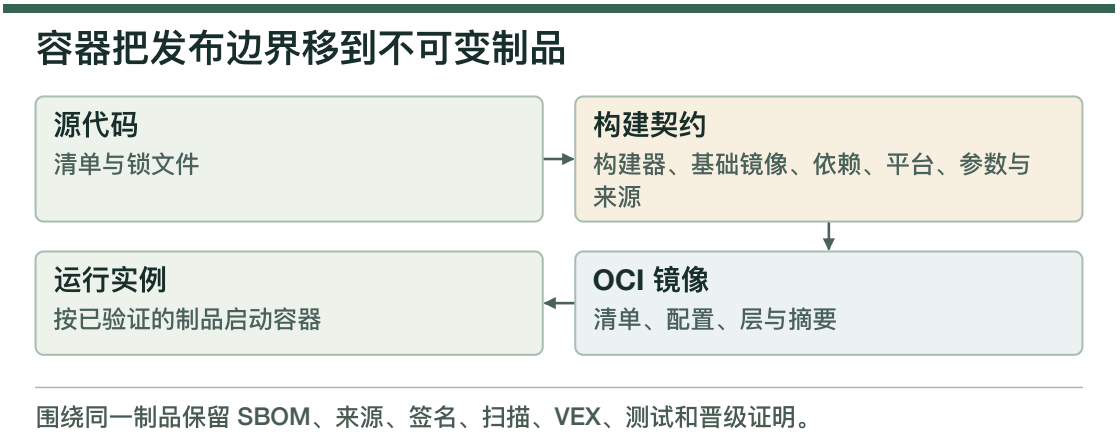


图 10-1 容器化把发布边界从已配置机器移到内容寻址工件及其证据。

表 10-1 容器化分离过去经常隐藏在服务器中的工件。

对象	主要责任	必须保持明确的内容
源代码修订	定义经过审查的应用意图。	仓库、提交、分支策略、依赖声明和构建定义。
构建环境	把源代码与材料转换为工件。	构建器身份、工具链、网络访问、参数、机密、平台和隔离。
OCI 镜像	打包运行时文件系统和配置。	摘要、清单、配置、层、架构、入口点、标签和媒体类型。
注册表	存储并分发镜像及相关工件。	身份、网络、仓库作用域、保留、复制、证据发现和审计。
运行时	实例化并约束容器进程。	镜像摘要、策略决策、运行时身份、文件系统可变性、资源和主机边界。
发布记录	解释工件为何以及向何处晋升。	审批、源环境、目标环境、策略版本、时间戳和回滚主体。

镜像与容器的区别因此属于架构层面。镜像是存储的工件图；容器是在特定运行时和主机策略下从该工件创建的运行时实例。签名镜像不等于签名运行进程，扫描镜像也不会检查每个运行时状态。反过来，运行时强化无法修复未知或不可信构建历史。供应链控制与运行时控制互为补充。

10.2 OCI 镜像、索引与分发契约

Open Container Initiative 标准化三个相关关注点：镜像如何表示、低级运行时如何执行容器包，以及内容如何通过注册表分发。镜像和分发规范是本章重点，因为它们定义云平台使用的可移植工件与注册表数据平面契约。

OCI 镜像不是单个归档，而是描述符图。描述符按媒体类型、摘要和大小标识内容，还可携带注解与目标平台。镜像清单指向镜像配置和一个或多个层 Blob；镜像配置包含运行时默认值和文件系统元数据；镜像索引（相邻生态中也称清单列表）可指向适用于不同操作系统与处理器架构的多个清单。

OCI 镜像的内容寻址图

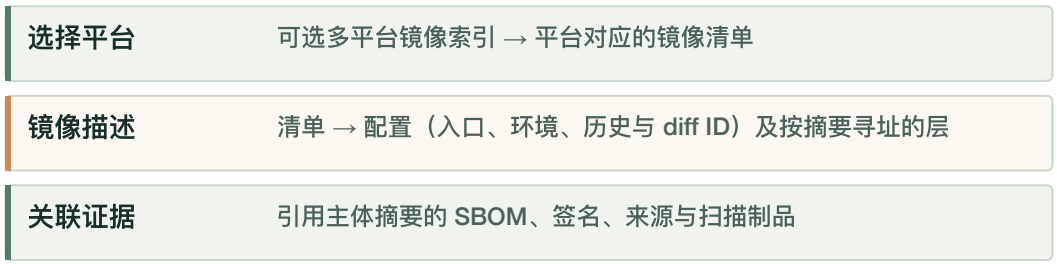


图 10-2 OCI 镜像是由描述符与内容寻址对象构成的图。

表 10-2 核心 OCI 对象及其架构角色。

OCI 对象	角色	治理关注点
描述符	把媒体类型、摘要、大小、注解和可选平台与内容绑定。	摘要验证、受支持算法、媒体类型处理和元数据信任。
镜像清单	描述一项可运行镜像配置及其有序层。	签名、扫描、SBOM、晋升和运行时固定使用的稳定主体。
镜像配置	定义运行时默认值和根文件系统历史。	入口点、命令、用户、环境、架构、创建元数据和历史泄漏。
层 Blob	携带压缩的文件系统变更集。	机密泄露、包清单、缓存复用、去重和删除语义。
镜像索引	通常按平台选择一个或多个清单。	多平台完整性、逐平台证据以及索引与清单的签名作用域。
工件清单 / 引用类型	表示非镜像 OCI 工件或引用某主体的证据。	工件类型、可发现性、主体绑定、保留和引用项晋升。
分发 API	标准化推送、拉取、清单、Blob、标签和引用项操作。	身份验证、授权、限制、网络暴露、兼容性和审计。

内容寻址意味着标识符由字节派生。内容改变，摘要也改变。注册表可以去重共享 Blob，使用者可以验证收到的内容。这一属性为供应链提供稳定主体。即使标签、发布通道和部署环境变化，证据仍可引用确切摘要。

OCI Image and Distribution Specification 1.1 引入标准主体关系和 Referrers API。签名、SBOM、来源声明、扫描结果或其他工件可以把镜像清单标识为主体。使用者随后可查询注册表中引用该摘要的工件。结果不只是仓库中的附加文件，而是关系能够经受标签移动的证据图。

10.3 标签、摘要、版本与晋升

标签便于人员和自动化使用，但属于可变引用。除非注册表强制不可变，否则 production、stable、v1，甚至 1.4.2 标签都可移动。摘要是不可变内容身份；语义版本描述发布意图；部署修订记录平台特定推出；晋升记录解释已构建工件为何在环境间移动。混淆这些概念会削弱回滚与审计。

表 10-3 发布标识符回答不同问题。

标识符	回答的问题	可变性与用途
仓库名称	这是哪个工件系列?	长期命名空间；不是版本。
标签	此通道或标签当前应解析到哪个工件?	通常可变；适合发现与工作流，不宜作为唯一生产身份。
摘要	预期的是哪一组确切清单字节?	不可变内容身份；首选用于签名、验证、晋升和运行时固定。
语义版本	该工件代表什么生产者可见发布意图?	人员契约；应映射到摘要并防止重新分配。
部署修订	哪个平台配置正在处理流量?	平台资源；可组合镜像摘要、配置、身份和扩缩设置。
晋升记录	为何批准此摘要进入下一个环境?	仅追加证据；应标识来源、目标、策略、审批者和主体摘要。

核心论点。 只构建一次，并晋升同一摘要。即使源提交相同，分别为测试和生产重建也会产生不同工件，从而破坏最强形式的发布等价性。

晋升可以把镜像及其引用项复制到另一个注册表或区域，为同一摘要添加受保护环境标签，或更新部署期望状态使其引用已批准摘要。机制可以变化，但主体不应变化。若工件重新构建，新摘要需要新的来源、扫描、签名、审批和验证。

多平台镜像还引入一项决策：签名并验证索引、每个平台清单，还是两者都做。索引为跨平台提供一个发布身份，而每个平台清单代表实际运行时字节。策略应规定预期作用域，并确保运行时选择的平台具备所需证据。

10.4 把构建作为受治理供应链契约

构建成功只证明构建器产生了输出，不能证明源代码经过审查、依赖项符合预期、构建器已隔离、基础镜像获批准、机密受保护或结果可重现。因此，构建必须建模为控制平面契约，而非本地命令。

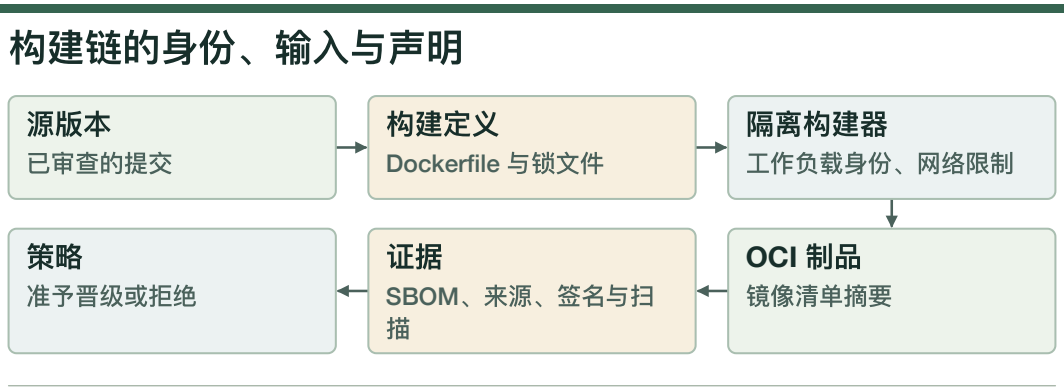


图 10-5 构建路径是一条身份、输入、工件、声明和策略决策链。

表 10-4 决定构建结果的输入与条件。

构建要素	示例	隐式时的风险
源代码身份	仓库 URI、提交摘要、分支、子模块。	未经审查或被替换的源代码可能进入发布。
构建定义	Dockerfile、构建脚本、buildpack、Bake 文件、工作流。	本地或已修改指令会生成未记录工件。
基础镜像	运行时与构建阶段的仓库和摘要。	可变标签或受损基础镜像会悄然改变工件。
依赖项	包锁定文件、包仓库、下载工具。	解析漂移和依赖混淆会破坏可重复性。
构建平台	构建器实现、版本、操作系统、架构、隔离。	特权或受损构建器可更改输出和证据。
参数	构建参数、目标阶段、功能标志、平台列表。	意外参数会生成有效但未授权的变体。
网络访问	允许的包注册表、镜像源、时间源、外部下载。	无界网络访问使材料捕获和封闭性难以实现。
机密与身份	注册表令牌、签名身份、包凭据。	机密泄露到层、日志、缓存、来源或不可信步骤。
输出契约	镜像名称、摘要、SBOM、来源、签名、测试结果。	流水线无法确定必须使用和验证什么。

10.4.1 可重复性、可重现性、封闭性与隔离

这些属性彼此相关但不可互换。可重复构建可由同一过程再次运行；可重现构建从相同声明输入生成等价输出，通常在另一可信系统上完成；封闭构建通过受控、声明的通道获得全部输入，不依赖未声明外部状态；隔离构建防止某次构建或租户影响另一构建。组织可

能无法立即实现全部属性，但应说明其来源声明支持哪些保证。

表 10-5 构建属性及其所需证据。

属性	含义	有用证据
可重复	可按同一预期配置再次调用构建过程。	版本化工作流、工具版本、参数记录、构建日志。
可重现	根据同一输入的独立构建产生等价输出。	源代码和材料摘要、确定性配置、比较结果。
封闭	只有声明输入影响构建。	网络策略、材料列表、依赖镜像源、沙盒或运行时跟踪。
隔离	并发或先前工作负载无法修改构建或其证据。	临时构建器、租户边界、不可变工作进程镜像、受保护签名身份。
已验证身份	生产者可标识构建器和证据发行者。	工作负载身份、签名来源、证书或透明度证据。
符合策略	构建满足预期源代码、构建器、参数和依赖规则。	验证器决策、策略版本、例外记录、主体摘要。

SLSA 表达逐步增强的构建来源和构建平台要求。其价值不在标签本身，而在验证模型：使用者把主体摘要的来源与有关源代码、构建器、参数和材料的预期比较。只生成但从不验证的来源信息，只是存储的元数据。

10.5 构建最小且可维护的镜像

多阶段构建把编译依赖与运行时镜像分离。BuildKit 可以并行化阶段、复用缓存、面向多个平台，并生成来源和 SBOM 证明。这些能力支持小型、可重复镜像，但安全结果仍取决于基础镜像策略、包选择、用户身份、文件系统权限和补丁节奏。

表 10-6 镜像强化决策及其权衡。

决策	收益	权衡或注意事项
多阶段构建	从运行时移除编译器、包管理器和仅构建文件。	从构建阶段复制的文件仍可能包含机密或不需要的依赖项。
最小基础镜像	减少包、攻击面、传输时间和扫描噪声。	可能移除运维所需 Shell、证书、时区或诊断工具。
以非 **root** 身份运行	降低容器内权限并限制常见提权路径。	可能需要更改文件所有权、端口、卷权限和旧式软件。
只读根文件系统	阻止许多未经授权的写入与持久化技术。	应用需要为临时与持久状态提供明确可写挂载。

决策	收益	权衡或注意事项
固定基础摘要	使构建输入明确并防止标签漂移。	组织必须监控并有意更新固定的基础镜像。
锁定依赖项	把包解析约束到经过审查的版本。	漏洞出现时必须更新、验证锁定文件并重新构建。
运行状况终结点	提供应用特定的就绪或存活证据。	肤浅检查可能在依赖项或业务功能失败时仍报告健康。
调试镜像分离	保持生产镜像最小，同时在其他位置保留故障排除工具。	操作员需要受控方法来附加临时调试工具。

```
# 语法有意省略固定工具版本；生产环境应固定批准摘要。
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ManagedBookApi.csproj .
RUN dotnet restore --locked-mode
COPY . .
RUN dotnet publish -c Release -o /out /p:UseAppHost=false

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS runtime
WORKDIR /app
ENV ASPNETCORE_URLS=http://+:8080
COPY --from=build /out .
USER $APP_UID
EXPOSE 8080
ENTRYPOINT ["dotnet", "ManagedBookApi.dll"]
```

对于生产 Dockerfile，应把基础标签替换为组织批准的摘要，记录补丁与重建策略，并验证所选基础镜像定义了 USER 预期的非 root 用户。

体积小不等于安全。只含一个静态链接二进制文件的镜像仍可能包含易受攻击的库、不安全配置或恶意代码。较大镜像可以维护良好且约束正确。大小是运维信号；信任则是由组成、来源、审查、策略和运行时控制支撑的声明。

10.6 注册表作为云控制平面与工件数据平面

托管注册表通常被描述为镜像存储，但其云资源模型更广。控制平面创建注册表实例、选择区域和服务层级、配置身份模式与网络边界，并管理保留、复制、诊断和成本；数据平面则推送、拉取、解析和发现仓库工件。



图 10-4 注册表管理与仓库操作属于不同平面和身份。

表 10-7 注册表控制平面对象及其运维用途。

控制平面对象	用途	证据与策略
注册表实例	定义区域服务、SKU、终结点、加密、身份模式和资源所有权。	资源 ID、配置历史、活动日志、成本作用域、服务运行状况。
命名空间 / 仓库策略	划分工件系列、团队或环境，并约束操作。	角色分配、ABAC 条件、仓库属性、审计事件。
网络配置	控制公共访问、专用终结点、VPC/VNet 可达性、允许列表和受信服务。	终结点状态、DNS、连接记录、流量或防火墙证据。
保留 / 清理	依据规定规则移除未标记或过时工件。	策略版本、试运行结果、删除日志、受保护发布例外。
复制 / 同步	把工件放在使用者附近，或提供区域恢复。	源与目标摘要相等性、复制状态、滞后、故障事件。
构建或任务资源	执行服务器端构建、基础镜像触发器或计划维护。	任务身份、来源、运行记录、日志、输出摘要、来源证明。
诊断设置	导出控制平面和数据平面活动，以供监控与审计。	日志目标、类别、保留，以及与身份和摘要的关联。
签名 / 信任集成	提供或连接签名密钥、证书、托管签名和验证策略。	签名者身份、证书链、时间戳、信任策略、验证结果。

10.6.1 身份：人员、构建器、签名者与运行时

除非有明确理由，同一身份不应同时构建、签名、审批和部署。开发者需要发现和故障排除访问；构建器需要读取源代码与基础镜像，并写入范围受限仓库；签名者需要调用受保护密钥或托管签名服务，但不一定需要更改镜像；运行时身份只需拉取所用仓库，最好仅限这些仓库；安全与发布团队需要访问证据，而无需广泛工件变更权限。

表 10-8 按参与者划分的注册表最小权限角色。

参与者	所需能力	避免授予
开发者	读取元数据、拉取批准镜像，并在需要时只向开发仓库推送。	注册表管理、生产删除、不受限制的签名密钥。
CI 构建器	推送清单和层、附加生成的证据、读取批准的基础镜像。	人员交互凭据、生产晋升权限、密钥导出。
签名者	使用受保护身份和密钥签署指定摘要。	在没有独立策略时重建或重新标记主体的能力。
发布晋升者	复制或重新标记已验证摘要，并记录审批。	源代码修改或签名密钥管理。
运行时 / 编排器	拉取确切批准摘要，并获取验证所需证据。	除非确有需要，否则不授予推送、删除、标签变更或宽泛目录访问。
安全扫描器	读取主体清单与层；获授权时附加扫描证据。	标签变更、删除、无关仓库访问。
注册表操作员	管理服务配置、网络、保留和诊断。	在控制平面与数据平面角色分离时隐式获得仓库数据访问。

Azure Container Registry 现在公开启用 ABAC 的仓库权限角色分配模式。在该模式下，Container Registry Repository Reader、Writer 和 Contributor 角色取代旧版 AcrPull、AcrPush、AcrDelete 的仓库数据操作行为。Contributor 等管理角色可以管理注册表资源，却不会自动获得仓库读写权限。这正是本书前文建立的控制平面/数据平面区分。

10.6.2 网络、保留、复制与成本

私有连接可以阻止注册表数据流量经过公共终结点，但不能替代仓库授权或签名验证。复制提升位置接近性和韧性，却会扩大必须保留同一摘要与证据图的区域和策略数量。保留会降低过时工件的存储成本与风险，但若未一体化处理标签和引用项，可能意外删除证据或回滚主体。

表 10-9 注册表经济性及其工程控制。

成本驱动因素	典型原因	控制
存储的层和工件	未使用标签、重复变体、保留的引用项、大型模型或调试镜像。	保留规则、仓库清单、层复用、发布保护、工件类别策略。
数据传输	跨区域拉取、公共出口、重复节点扩容、远程构建上下文。	区域复制、私有访问、缓存、镜像最小化、工作负载放置。
请求与吞吐容量	大型集群、许多小层、重复标签解析、证据查询。	固定摘要、镜像预拉取、请求规划、合适注册表层级、受支持时延迟加载。

成本驱动因素	典型原因	控制
扫描与安全服务	自动重新扫描、高级扫描器、报告保留。	基于风险的作用域、扫描频率、证据保留、VEX、过期结果策略。
签名与密钥服务	证书签发、HSM 或 Key Vault 操作、时间戳、托管签名。	密钥层次、短期身份、在安全时批处理、显式保留和验证策略。
复制与恢复	多个区域副本与同步工件图。	把副本放在合理使用者附近；验证引用项与策略随主体移动。

10.7 证据：SBOM、来源、签名、扫描、VEX 与测试

软件供应链证据经常被当作每项工件的安全证书。更准确的做法是把它们视为关于主体摘要的类型化声明。SBOM 声明组成；来源声明工件如何生成；签名把身份绑定到主体描述符；漏洞报告记录特定时间的扫描结果；VEX 文档说明已知漏洞是否被认为影响产品；测试证据记录执行了什么以及什么通过。没有任何一项声明能够单独证明软件安全。



图 10-3 OCI 引用项可围绕一个不可变主体摘要组织供应链证据。

表 10-10 证据类型回答不同问题，也有不同局限。

证据类型	回答的问题	不能证明什么
SBOM	报告了哪些组件、包、文件、许可证或关系？	不能证明发现了每个组件、组件安全或 SBOM 与运行时状态一致。
构建来源	哪些源代码、材料、构建器、参数和过程生成了主体？	不能证明源代码无害、构建器应受信任或输出没有漏洞。

证据类型	回答的问题	不能证明什么
签名	可信身份是否签署了此确切描述符，且签名是否符合策略？	不能证明签名者正确审查了工件或工件安全。
漏洞扫描	扫描器在此时对此工件报告了哪些已知发现？	不能证明未来发现、可利用性、隐藏代码、配置错误或运行时暴露。
VEX	生产者或分析人员是否认为特定漏洞适用、不受影响、已修复或正在调查？	没有可信发行者、依据和审查时，不能证明断言正确。
测试证明	哪个测试套件针对哪个主体运行，结果如何？	不能证明测试充分、独立或能代表生产环境。
晋升证明	哪个已验证摘要依据哪项策略获准进入目标环境？	不能证明目标实际部署了它或运行时配置持续合规。

10.7.1 SBOM 格式与生命周期

SPDX 和 CycloneDX 是广泛使用的机器可读软件物料清单标准。SPDX 是 ISO 认可的开放标准，其配置文件超越基本包清单；CycloneDX 是全栈 BOM 标准，可表示软件、服务、硬件、密码学和机器学习工件。工具与生态兼容性可能要求多种输出格式，但转换不应被误认为信息等价。

表 10-11 SBOM 设计决策。

决策	要记录的问题
生成阶段	SBOM 是从源代码依赖、构建输出、最终镜像文件系统，还是三者生成？
主体	SBOM 引用镜像索引、平台清单、包、二进制文件、模型工件还是发布包？
深度	是否包含传递包、文件、操作系统组件、许可证、哈希和关系？
完整性	生成器无法识别哪些生态或文件类型？
新鲜度	SBOM 是否为不可变组成证据，而漏洞数据另行更新？
分发	SBOM 是嵌入、作为 OCI 引用项附加、存入证据系统，还是三者并用？
访问	使用者能否在不取得工件变更权限的情况下获取 SBOM？
保留	每个受支持版本和事件窗口期间，SBOM 与相关证据是否持续可用？

BuildKit 可以在 docker buildx build 期间生成 SBOM 和来源证明。输出推送到兼容注册表时，证明可附加到镜像或镜像索引。组织仍须检查生成的谓词及版本、机密或敏感参数是否可能泄露到来源中，以及下游工具能否一致发现附件。

10.7.2 来源只有经过验证才有用

in-toto 证明声明把主体摘要绑定到谓词类型和谓词；SLSA 来源使用该结构描述构建。验证器应把声明与预期比较：可信构建器身份、批准的源仓库与修订、允许参数、所需材料和完整性。仅把来源存到镜像旁不会强制实施这些预期。

来源也有隐私影响。完整构建参数和环境细节可能暴露内部路径、仓库名称、构建参数或基础设施。发布策略应定义哪些证据对外可见、哪些留在组织内部，以及部署门禁要求哪些最低字段。

10.8 漏洞管理是生命周期，而非一道门禁

扫描结果是由特定扫描器、数据库、包识别方法和策略在某一时间生成的视图。即使镜像字节完全不变，仍会披露新漏洞，严重性和可利用性评估也会变化。因此，供应链控制平面必须支持工件投入生产后的重新扫描、分类、修复、重建、替换和退役。

表 10-12 漏洞管理阶段。

阶段	控制	证据
基础镜像选择	使用按摘要固定、获批准且受维护的基础镜像。	基础清单、支持策略、来源、摘要、时间。
构建时扫描	发布前检测易受攻击的依赖项。	扫描器版本、数据库时间戳、主体摘要、发现、策略结果。
注册表扫描	评估存储工件，并在漏洞数据变化时重新扫描。	注册表扫描报告、重新扫描时间戳、仓库和摘要。
分类	结合严重性、可达性、可利用性、工作负载暴露和业务关键性。	决策所有者、依据、截止日期、VEX 或例外。
修复	更新基础镜像或依赖项，重建、复测、重新签名并重新晋升。	新摘要、差异 SBOM、新来源、验证和推出记录。
运行时清单	识别受影响摘要在哪里运行。	集群或平台清单、工作负载所有者、环境、暴露。
退役	从活动通道移除易受攻击或不受支持工件，最终从注册表删除。	弃用通知、替换摘要、保留决策、删除证据。

仅按严重性制定部署策略并不完整。执行路径中不存在的组件内的关键漏洞，与可从公共终结点触达的高严重性漏洞，可能需要不同响应。VEX 可以传达适用性评估，但 VEX 断言本身也是需要发行者、状态、依据、时间戳和审查过程的证据。

最安全的修复模式通常是重建并替换，而不是就地变更已发布镜像。重建产生新摘要与新证据图。发布系统随后可使用相同生命周期契约验证、晋升、部署、观察新摘要并退役旧摘要。

10.9 签名与信任策略

签名把签名者身份绑定到工件描述符摘要。验证判断签名在密码学上是否有效，并在信任策略下是否可接受。策略不可或缺：如果没有预期身份、作用域、证书链、时间戳和验证级别，来自非预期签名者的有效签名仍可能被拒绝，甚至被错误接受。

表 10-13 OCI 工件的常见签名方式。

方式	信任模型	运维特征
Notation 加组织管理的证书	X.509 证书链和显式 Notary Project 信任策略。	组织控制强；需要签发、存储、轮换、吊销、时间戳和信任存储分发。
Notation 加托管 **Artifact Signing**	通过托管签名服务签发并在配置的信任下验证的短期证书。	减少证书生命周期工作；仍需身份策略、时间戳验证、作用域和使用者配置。
Cosign 无密钥签名	OIDC 身份、临时证书、透明度日志和身份约束。	适合 CI 联合与开放生态；依赖发行者、身份匹配、透明度与可用性策略。
Cosign 或 **Notation** 加受保护密钥	与受保护本地、HSM、KMS 或 Key Vault 密钥关联的公钥或证书。	适合私有环境；密钥保管、轮换、分发和吊销仍是组织职责。

应按摘要签名，而不是按标签。即使发布通道标签移动，签名主体仍须稳定。时间戳让验证器确定签名创建于签名证书有效期间，对短期证书或长期支持版本尤其重要。轮换应在移除旧信任前引入新可信签名者；吊销与泄露响应应经过测试，而非假定可用。

表 10-14 信任策略应明确以下决策。

策略要素	示例问题
主体作用域	覆盖哪个注册表、仓库、工件类型和摘要？
可信身份	哪个 X.509 主题、OIDC 身份、组织或密钥获准签名？
验证级别	是否要求完整性、真实性、到期、吊销和时间戳检查？
证据集	签名是否足够，还是还要求来源、SBOM、扫描和测试证明？

策略要素	示例问题
新鲜度	扫描、审批和漏洞数据库最长可以多旧?
环境	生产环境是否要求比开发更强的签名者或策略?
例外	谁可批准例外、持续多久、需要哪些补偿控制?
失败动作	审计、警告、隔离、拒绝晋升、拒绝准入或终止运行时?

10.10 从源代码到运行时的策略门禁

软件供应链由一系列门禁保护，而非单一安全产品。源代码控制确定谁能更改代码和构建定义；构建控制确定谁能生成工件与证据；注册表控制确定谁能发布、变更、发现、复制和删除；晋升控制确定哪个摘要可进入环境；准入控制确定运行时可实例化什么；运行时控制确定实际运行什么，以及如何处理新发现风险。

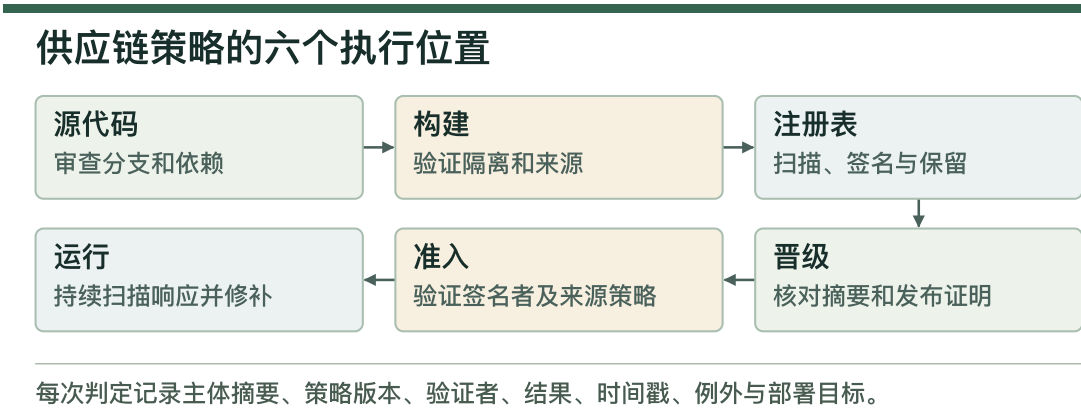


图 10-6 策略门禁保护工件生命周期的不同阶段。

表 10-15 供应链门禁及故障动作示例。

门禁	所需断言	典型失败动作
源代码合并	经过审查的变更、受保护分支、批准的依赖项、无暴露机密。	拒绝合并或要求额外审查。
构建发布	可信构建器、预期源修订、测试成功、已生成 SBOM 和来源。	不推送，或把工件标记为隔离。
注册表接受	允许的仓库、获授权生产者、扫描低于策略阈值、已附加签名。	在支持时拒绝推送、阻止拉取或阻止晋升。
环境晋升	同一已验证摘要、批准的证据、所有者和变更记录、没有过期例外。	把摘要留在较低环境并记录决策。

门禁	所需断言	典型失败动作
Kubernetes 或平台准入	固定摘要、可信签名、预期来源、允许的注册表、工作负载策略。	拒绝创建工作负载，或在分阶段推出中审计。
运行时响应	运行摘要清单、当前漏洞和暴露状态、受支持版本。	警报、隔离、缩容、替换或执行事件程序。

策略引擎应记录主体摘要、策略版本、验证器身份与软件版本、决策、时间戳、证据引用、目标环境和例外。该记录使拒绝可解释、批准可审计，也支持渐进实施：先观察，修复误报和缺失元数据，再把选定规则从审计切换到拒绝。

Kubernetes 建议避免 latest 等可变标签，并使用完整摘要获得稳定运行时身份。Pod 接受前，准入控制器可验证签名与证明。Azure 提供 Ratify 与 Azure Policy 模式在 AKS 上验证 Notary Project 签名；其他环境可使用 Sigstore policy-controller、Gatekeeper 集成、Kyverno 或提供商原生控制。实现方式不如契约重要：运行时必须依据当前信任策略验证确切主体。

10.11 AI 原生供应链超越镜像

AI 服务可以打包在容器中，却仍依赖单独分发的模型权重、分词器文件、适配器、量化参数、提示词、检索索引、安全配置、评估集和工具定义。若只有运行时镜像经过版本控制与签名，部署行为仍可能在没有相应软件发布的情况下改变。



图 10-7 AI 原生发布组合软件、模型、数据、配置、安全、评估与部署证据。

表 10-16 AI 原生工件及其所需声明。

工件	身份与证据要求
运行时镜像	摘要、SBOM、构建来源、签名、扫描、受支持平台和运行时策略。
模型权重	模型标识符、确切哈希或 OCI 摘要、来源、许可证、架构、来源证明、签名、评估作用域。
分词器 / 处理器	与模型耦合的版本和哈希；兼容性与来源证据。
适配器 / 微调	基础模型主体、训练数据快照、方法、超参数、生产者、评估和审批。
提示词 / 编排配置	版本、所有者、模型兼容性、工具架构、安全策略、评估结果、推出记录。
数据集或 RAG 语料库快照	来源、许可证、分类、血缘、选择规则、哈希、保留与删除义务。
安全策略	策略版本、阈值、提供商设置、评估覆盖率、审批者和生效环境。
评估集与结果	数据集版本、评估器版本、指标、阈值、模型与配置主体、成本和延迟。
部署包	镜像、模型、知识、提示词、工具、身份、区域和终结点的确切组合。

OCI 注册表可以存储任意合规工件与引用关系，因此可作为模型和相关证据的分发底座。腾讯云 TCR 目前记录了基于 OCI 的 AI 模型与技能托管，其他注册表即使没有模型特定体验，也支持通用 OCI 工件。架构不应假定每项 AI 工件都属于同一注册表；无论工件存在哪里，都应要求稳定身份、证据、访问控制、生命周期和部署时绑定。

NIST SP 800-218A 把安全软件开发实践扩展到 AI 模型开发。这强化了核心观点：模型开发和发布会增加新的供应链主体与风险，但不会消除对安全源代码、构建、来源、验证、受控分发和事件响应的需要。

10.12 Azure 参考控制平面

Azure Container Registry 是有用的参照，因为注册表、仓库权限、任务、私有连接、复制、工件、签名、扫描集成和准入验证仍然作为不同资源与操作可见。这并不意味着每个环境必须使用所有能力，而是演示如何把发布路径分解为可治理契约。

表 10-17 Azure 供应链控制平面资源。

能力	Azure 资源与工具	控制平面用途
注册表服务	Azure Container Registry 实例、SKU、终结点、可选区域副本。	定义所有权、区域、服务容量、可用性、加密、成本和生命周期。
仓库权限	Microsoft Entra RBAC；启用 ABAC 的 Repository Reader、Writer、Contributor 和 Catalog Lister 角色。	把注册表管理与仓库拉取、推送、元数据、删除和目录访问分离。
构建	Docker BuildKit/buildx、ACR Tasks、Azure Pipelines、GitHub Actions。	在工作负载身份下生成摘要、SBOM、来源、日志和可重复流水线证据。
OCI 证据图	ACR OCI 工件支持、ORAS、OCI Referrers API、az acr manifest list-referrers。	按主体摘要附加并发现签名、SBOM、扫描结果和其他证据。
签名	Notation 加 Key Vault 证书，或 Notation 加 Artifact Signing。	把可信发布者身份绑定到工件摘要，并管理证书或托管签名生命周期。
扫描	Microsoft Defender for Containers 和第三方或流水线扫描器。	发现已知漏洞并维护注册表/运行时风险证据。
准入	AKS 上的 Ratify 与 Azure Policy；可用时使用托管镜像完整性选项。	准入工作负载前验证签名和信任策略。
网络	Private Link、防火墙和受信服务设置、私有 DNS、区域终结点。	独立于仓库授权控制注册表可达性。
保留与恢复	保留、软删除、锁、仓库属性和清除任务。	在保留回滚主体与证据的同时控制过时工件。

能力	Azure 资源与工具	控制平面用途
审计与可观测性	Activity Log、诊断日志、任务运行、Defender 发现和策略决策。	关联身份、操作、摘要、策略、结果、目标和成本。

Azure CLI 清单命令、某些工件流式传输能力和选定完整性功能可能处于预览状态或取决于版本。生产使用前请验证当前命令状态、区域、SKU 和注册表加密约束。

对于新注册表，Azure 建议使用 RBAC Registry + ABAC Repository Permissions 模式。该模式中，旧版 AcrPull、AcrPush、AcrDelete 角色不再用于仓库访问；请使用启用 ABAC 的仓库角色。没有 ABAC 条件的角色会授予整个注册表范围的仓库访问，而条件可限制仓库名称。目录列表是独立权限。当流水线或集群看似拥有管理访问，却仍收到数据平面授权失败时，这些细节非常重要。

ACR 可以通过 OCI 引用存储供应链工件图。ORAS 可以附加和发现任意证据；Notation 创建引用镜像主体的签名；使用者配置信任存储和信任策略，再验证确切摘要。在 AKS 准入阶段，Ratify 与 Azure Policy 可拒绝签名不满足策略的工作负载。这是一条完整的生产者到使用者信任路径，而不是为合规展示创建的签名文件。

10.13 多云转译

各云提供商通过不同版本、资源层次、身份系统、扫描器、签名模型和 Kubernetes 集成公开相同工程意图。转译应保留这些差异。免费个人注册表可能适合实验，但缺少生产所需的隔离、SLA、网络控制、签名、扫描和复制。

表 10-18 多云注册表和供应链能力映射。

工程意图	Azure	Alibaba Cloud	Tencent Cloud	Volcengine
托管注册表	Azure Container Registry（ACR）	Container Registry（ACR），生产环境尤其使用企业版	Tencent Container Registry（TCR）企业版	Container Registry（CR）
OCI 工件支持	镜像和任意 OCI 工件；用于签名、SBOM 和扫描结果的引用项图	企业版在受支持实例上支持 OCI 工件和 OCI 1.1 引用类型 / Referrers API	Docker/OCI 工件、Helm Chart；当前文档还介绍 OCI AI 模型和技能托管	符合 OCI 的镜像和 Helm Chart；通用工件能力应按版本和工具验证
身份与作用域	Entra RBAC 和 ABAC 仓库角色；托管身份与服务主体	RAM、实例/命名空间/仓库授权，受支持企业场景中的无凭据拉取	CAM、实例/命名空间/仓库策略、服务账户和 TKE 集成	IAM 授权、注册表凭据和 VKE 凭据控制器集成
私有访问	Private Link、防火墙、受信服务和私有 DNS	VPC 访问控制、允许列表和企业网络隔离	VPC 私有访问控制与 TKE 私有无密码拉取集成	VPC 访问配额、终结点控制和通过 VKE 集成的私有拉取

工程意图	Azure	Alibaba Cloud	Tencent Cloud	Volcengine
扫描	Defender for Containers 和流水线/第三方扫描器	按版本提供多引擎工件扫描、修复指导和风险阻止	按版本提供企业镜像扫描、自动扫描和部署阻止	多维镜像扫描和构建时或平台安全集成
签名 / 验证	Notation 加 Key Vault 或 Artifact Signing；AKS 上的 Ratify/Azure Policy	企业签名与验证，包括 Cosign 兼容工作流和交付链控制	使用 KMS 的企业镜像签名与验证；TKE 集成和版本约束	不假设存在完全等价的托管签名；在受支持时使用 OCI 兼容外部签名并验证当前产品能力
复制 / 分发	地理复制、导入、缓存和连接注册表选项	按版本提供全局同步、P2P 和按需分发	按版本提供跨区域与跨账户同步、P2P 和延迟加载	跨区域分发、代理缓存和加速能力
生命周期与证据	保留、软删除、锁、任务运行、Activity Log 和诊断	清理、不可变标签、ActionTrail，以及构建/扫描/签名状态的 EventBridge 通知	保留规则、审计日志、触发器和同步历史	生命周期管理、审计和监控能力；验证当前证据粒度

产品版本和区域可用性会变化。请把该映射视为工程意图转译，再查验当前官方文档中的签名模型、OCI 引用支持、扫描引擎、Kubernetes 准入集成、网络路径、配额、SLA 和计费。

10.14 实验 10：构建、推送、扫描、签名和验证容器镜像

本实验验证一项能力：生产者可以按摘要发布 OCI 镜像、附加构建证据、生成漏洞报告、用受保护密钥签名不可变主体、依据显式信任策略验证签名，并收集可复用证据包。它不把镜像部署到 Kubernetes；准入验证属于第 11 章。

10.14.1 目标、架构与成本边界

表 10-19 实验 10 契约。

项目	决策
能力目标	创建最小非 root 镜像，推送到 ACR，按摘要标识它，捕获 SBOM 和来源，扫描，用 Notation 与 Key Vault 签名，再依据严格信任策略验证。
主要资源	资源组、启用 ABAC 的 Standard ACR、启用 RBAC 的 Key Vault、代码签名证书、仓库、OCI 镜像和引用项工件。
所需工具	Azure CLI、带 Buildx/BuildKit 的 Docker、.NET 8 SDK、jq、Trivy 或批准的扫描器、Notation、notation-azure-kv 插件；ORAS 可用于独立检查引用项。

项目	决策
证据	注册表配置、角色分配、源提交或源哈希、镜像摘要、清单元数据、SBOM/来源引用项、扫描报告、签名列表、信任策略、成功验证和清理证明。
成本边界	ACR 与 Key Vault 会产生费用。使用一个小型注册表和一张证书，迅速完成，删除资源组，并移除本地信任文件。实验不要求 Defender。
未演示	生产 CA 治理、透明度服务、多区域晋升、准入拒绝、密钥泄露演练或持续重新扫描。

使用非生产订阅或沙盒。命令假设使用 Bash。工具版本会变化；请从官方来源安装当前受支持版本，并记录在 evidence/tool-versions.txt。签名证书在实验中自签名，不得视为生产公钥基础设施模式。

```
set -euo pipefail
export SUFFIX=$(openssl rand -hex 3)
export LOCATION=eastus
export RG=book-ch10-supply-$SUFFIX
export ACR=booksupply$SUFFIX
export KV=booksign$SUFFIX
export REPO=managed-book-api
export TAG=1.0.0
export CERT_NAME=book-lab-signer
export CERT_SUBJECT="CN=book-lab-signer,O=Cloud Services
Lab,L=Seattle,ST=WA,C=US"
mkdir -p evidence

az login
az account show --output table
export SUB_ID=$(az account show --query id -o tsv)
export USER_ID=$(az ad signed-in-user show --query id -o tsv)
az group create --name "$RG" --location "$LOCATION" --tags   owner=book-lab
workload=supply-chain lifecycle=temporary
```

10.14.2 创建注册表与签名边界

以启用 ABAC 的权限模式创建 ACR，并禁用管理员账户。创建使用 Azure RBAC 授权的 Key Vault。只分配实验所需的仓库写入权限、证书与密码学角色。为简化命令，示例在注册表作用域分配仓库角色；生产工作流应为具体仓库添加 ABAC 条件。

```
az acr create   --resource-group "$RG"   --name "$ACR"   --location "$LOCATION"
--sku Standard   --admin-enabled false   --role-assignment-mode rbac-abac

export ACR_ID=$(az acr show -g "$RG" -n "$ACR" --query id -o tsv)
```

```
export REGISTRY=$(az acr show -g "$RG" -n "$ACR" --query loginServer -o tsv)

az keyvault create   --resource-group "$RG"   --name "$KV"   --location
"$LOCATION"   --enable-rbac-authorization true
export KV_ID=$(az keyvault show -g "$RG" -n "$KV" --query id -o tsv)

az role assignment create   --assignee-object-id "$USER_ID"   --assignee-
principal-type User   --role "Container Registry Repository Writer"   --scope
"$ACR_ID"

az role assignment create   --assignee-object-id "$USER_ID"   --assignee-
principal-type User   --role "Key Vault Certificates Officer"   --scope "$KV_ID"
az role assignment create   --assignee-object-id "$USER_ID"   --assignee-
principal-type User   --role "Key Vault Crypto User"   --scope "$KV_ID"

az acr show -g "$RG" -n "$ACR" > evidence/acr-resource.json
az role assignment list --scope "$ACR_ID" > evidence/acr-role-assignments.json
az role assignment list --scope "$KV_ID" > evidence/keyvault-role-
assignments.json
```

Azure 角色分配可能需要几分钟传播。如果身份验证成功，但推送或签名返回 403，请等待后重试。更改角色分配权限模式后刷新 ACR 凭据。

10.14.3 创建最小非 root API

```
mkdir -p src && cd src
dotnet new web -n ManagedBookApi --framework net8.0
cd ManagedBookApi
cat > Program.cs <<'CS'
var app = WebApplication.Create(args);
app.MapGet("/", () => Results.Ok(new { service = "managed-book-api", version =
"1.0.0" }));
app.MapGet("/health", () => Results.Ok(new { status = "healthy" }));
app.Run();
CS

dotnet restore --use-lock-file
cat > Dockerfile <<'DOCKER'
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ManagedBookApi.csproj packages.lock.json ./
RUN dotnet restore --locked-mode
COPY . .
RUN dotnet publish -c Release -o /out /p:UseAppHost=false

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS runtime
WORKDIR /app
```

```
ENV ASPNETCORE_URLS=http://+:8080
COPY --from=build /out .
USER $APP_UID
EXPOSE 8080
ENTRYPOINT ["dotnet", "ManagedBookApi.dll"]
DOCKER

cat > .dockerignore <<'EOF'
bin/
obj/
evidence/
.git/
EOF
cd ../..
```

真实发布应按批准摘要固定两个基础镜像，并记录更新程序。实验保留可读的 Microsoft 基础标签，便于重现工作流；证据包应明确记录此限制。

10.14.4 只构建一次、推送并捕获摘要

使用已登录 Entra 身份进行身份验证，必要时创建 Buildx 构建器，并在请求 SBOM 和完整来源证明的同时推送镜像。输出推送到注册表时，证明比仅加载到本地镜像存储更有用。

```
az acr login --name "$ACR"
docker buildx create --name book-supply-builder --use 2>/dev/null ||   docker
buildx use book-supply-builder
docker buildx inspect --bootstrap

export IMAGE_TAGGED="$REGISTRY/$REPO:$TAG"
docker buildx build   --platform linux/amd64   --sbom=true   --
provenance=mode=max   --tag "$IMAGE_TAGGED"   --push   ./src/ManagedBookApi

export DIGEST=$(az acr manifest show-metadata   --registry "$ACR"   --name
"$REPO:$TAG"   --query digest -o tsv)
export IMAGE="$REGISTRY/$REPO@$DIGEST"
printf '%s
' "$IMAGE" | tee evidence/image-subject.txt
az acr manifest show-metadata -r "$ACR" -n "$REPO@$DIGEST"   >
evidence/manifest-metadata.json
az acr manifest list-referrers -r "$ACR" -n "$REPO@$DIGEST"   >
evidence/referrers-before-signing.json
```

az acr manifest 命令组目前标记为预览。若语法变化，请使用 oras discover --format json "$IMAGE" 或注册表 OCI Referrers API，并在证据包中记录工具版本。

检查引用项输出。根据 Buildx、注册表和多平台清单行为，SBOM 与来源可能引用镜像索引或某个平台清单。记录实际主体，不要假定一个证据对象自动覆盖每个平台变体。

10.14.5 扫描不可变主体

对摘要而非可变标签使用批准扫描器。以下示例使用 Trivy，因为它可以不依赖付费云扫描计划独立运行。可替换为 Docker Scout 或其他企业扫描器，但证据包必须记录扫描器、版本、数据库更新时间、主体摘要和策略阈值。

```
trivy --version | tee evidence/trivy-version.txt
trivy image   --scanners vuln,secret   --format json   --output
evidence/scan.json   "$IMAGE"

# Produce a human-readable summary without changing the subject.
trivy image --severity HIGH,CRITICAL "$IMAGE"   | tee evidence/scan-summary.txt
```

发现结果并不自动代表可利用，干净报告也不能证明不存在漏洞。把结果记录为有时间边界的扫描声明。若策略失败，应从更新材料重建并生成新摘要，而不是编辑已发布镜像。

10.14.6 创建实验签名证书

```
cat > evidence/certificate-policy.json <<EOF
{
  "issuerParameters": { "name": "Self" },
  "keyProperties": {
    "exportable": false,
    "keySize": 2048,
    "keyType": "RSA",
    "reuseKey": false
  },
  "secretProperties": { "contentType": "application/x-pem-file" },
  "x509CertificateProperties": {
    "ekus": [ "1.3.6.1.5.5.7.3.3" ],
    "keyUsage": [ "digitalSignature" ],
    "subject": "$CERT_SUBJECT",
    "validityInMonths": 3
  }
}
EOF

az keyvault certificate create   --vault-name "$KV"   --name "$CERT_NAME"   --
policy @evidence/certificate-policy.json   > evidence/certificate-create.json

export KEY_ID=$(az keyvault certificate show   --vault-name "$KV"   --name
```

```
"$CERT_NAME"   --query kid -o tsv)
printf '%s
' "$KEY_ID" > evidence/signing-key-id.txt
```

私钥在 Key Vault 中保持不可导出。Notation 插件通过当前 Azure 身份请求签名操作。在生产中，证书主题、发行者、有效期、时间戳机构、角色分配和职责分离必须独立于镜像流水线进行治理。

10.14.7 按摘要签名并检查引用项图

```
notation plugin ls | tee evidence/notation-plugins.txt
notation version | tee evidence/notation-version.txt

notation sign   --signature-format cose   --id "$KEY_ID"   --plugin azure-kv   -
-plugin-config self_signed=true   --plugin-config credential_type=azurecli
"$IMAGE"

notation ls "$IMAGE" | tee evidence/notation-list.txt
az acr manifest list-referrers -r "$ACR" -n "$REPO@$DIGEST"   >
evidence/referrers-after-signing.json
```

签名是指向不可变主体的引用项。列出图后应显示构建证明和 Notary Project 签名，但确切媒体类型与附件结构取决于工具版本。如果注册表或加密配置要求使用引用项标签方案而不是 Referrers API，请记录该实现细节；信任声明仍必须绑定摘要。

10.14.8 创建信任存储并验证

```
export CERT_PATH=./evidence/${CERT_NAME}.pem
az keyvault certificate download   --vault-name "$KV"   --name "$CERT_NAME"   --
file "$CERT_PATH"

export STORE_TYPE=ca
export STORE_NAME=book-lab
notation cert add --type "$STORE_TYPE" --store "$STORE_NAME" "$CERT_PATH"

cat > evidence/trustpolicy.json <<EOF
{
  "version": "1.0",
  "trustPolicies": [
    {
      "name": "book-lab-images",
      "registryScopes": [ "$REGISTRY/$REPO" ],
      "signatureVerification": { "level": "strict" },
      "trustStores": [ "$STORE_TYPE:$STORE_NAME" ],
```

```
      "trustedIdentities": [ "x509.subject: $CERT_SUBJECT" ]
    }
  ]
}
EOF

notation policy import evidence/trustpolicy.json
notation policy show | tee evidence/trust-policy-active.txt
notation verify "$IMAGE" | tee evidence/verification.txt
```

验证应返回经过验证的摘要。把信任策略与验证输出一并保存。只有因本地客户端信任所有身份才通过验证的签名，不是有意义的证据；策略必须标识注册表作用域和预期签名者。

10.14.9 组装证据包

表 10-20 实验 10 所需证据包。

文件或记录	回答的问题
tool-versions.txt	哪些 CLI、构建器、扫描器、签名和注册表工具生成或检查了证据？
acr-resource.json	哪个注册表资源、区域、SKU 和角色分配模式托管了工件？
角色分配文件	哪个主体可以写入仓库并调用签名密钥？
image-subject.txt / manifest-metadata.json	哪个确切摘要是发布主体？
referrers-before-signing.json	构建后存在哪些 SBOM 和来源附件？
scan.json / scan-summary.txt	所选扫描器在何时报告了什么？
certificate-create.json / signing-key-id.txt	使用了哪个受保护签名身份和证书版本？
notation-list.txt / referrers-after-signing.json	哪些签名和其他派生工件引用主体？
trustpolicy.json / verification.txt	信任哪个签名者和作用域，确切摘要是否验证成功？
cleanup.txt	实验后云端和本地签名资源是否退役？

```
{
  az --version
  docker version --format '{{.Client.Version}}'
  docker buildx version
  jq --version
  trivy --version | head -1
  notation version
  notation plugin ls
} > evidence/tool-versions.txt

sha256sum evidence/* > evidence/evidence-packet.sha256
ls -lh evidence
```

可选：使用另一个主体执行否定授权测试，该主体具有注册表管理 Reader 权限，但无仓库数据平面角色。预期结果是该主体可检查注册表资源，却无法拉取私有镜像。由于沙盒环境中的身份创建与凭据处理不同，此测试保持可选。

10.14.10 清理并证明退役

```
# Remove local Notation policy and trust material according to your workstation
policy.
notation policy show > evidence/trust-policy-before-cleanup.txt || true

az group delete --name "$RG" --yes --no-wait
az group wait --deleted --name "$RG"

if az group exists --name "$RG" | grep -q true; then
  echo "cleanup failed: resource group still exists" | tee evidence/cleanup.txt
  exit 1
else
  echo "cloud cleanup verified: resource group deleted" | tee
evidence/cleanup.txt
fi

docker buildx rm book-supply-builder 2>/dev/null || true
# Delete the lab trust store and imported policy using the supported Notation
workflow
# for the installed version, then securely remove local certificate and evidence
copies
# when they are no longer needed.
```

Key Vault 使用软删除，可能在恢复期内保留已删除保管库名称。在启用清除保护或组织保留的订阅中，实验不得绕过该策略。把保留的已删除状态记录为清理证据的一部分。

10.14.11 故障排除与解释

表 10-21 实验故障模式与诊断问题。

症状	可能原因或检查项
az acr login 成功，但推送返回 **403**	确认启用 ABAC 的模式、Repository Writer 角色、角色传播和刷新凭据。Contributor 只有控制平面访问权限。
构建已推送，但没有 **SBOM** 或来源引用项	确认 Buildx 驱动、--push、证明标志、主体作用域、注册表兼容性和 CLI 预览行为；使用 ORAS 检查。
扫描器无法拉取摘要	确认注册表身份验证和仓库 Reader/Writer 数据平面访问；使用 ACR 返回的同一登录服务器。
Key Vault 证书创建遭拒	确认 Certificates Officer 角色、RBAC 模式、订阅和传播。
Notation 签名遭拒	确认 Crypto User 角色、证书密钥 ID、插件安装和凭据类型。
Notation 验证标签，但摘要与发布记录不同	停止并验证显式摘要。标签可能已经移动。
签名存在，但验证拒绝身份	比较证书主题、注册表作用域、信任存储类型、信任策略和证书链。
清理后无法复用 **Key Vault** 名称	软删除保留可能占用名称；使用新后缀或遵循获授权清除策略。

成功结果不是“镜像安全”。结果更狭窄且可审计：特定摘要由记录的输入构建，发布到受治理注册表，伴有 SBOM 和来源证据，由具名工具在记录时间扫描，通过受保护密钥由预期身份签名，并依据严格策略验证。生产保障还需要源代码、构建平台、准入、运行时、事件和组织控制。

10.15 CloudDesk 复用

CloudDesk FinOps Agent 不应获得能够运行任意 Docker、Azure CLI 或注册表命令的通用 Shell 工具。本章生成范围受限、面向读取的能力，今后可由 Tool Broker 封装；需要变更时，再扩展为计划、审批、执行和验证阶段。

表 10-22 CloudDesk 能力移交。

能力	输入	输出与护栏
container.image.scan	提供商、注册表、仓库、摘要、扫描器策略和新鲜度要求。	规范化发现、扫描器元数据、证据 URI、陈旧程度、策略结果；只读且绑定摘要。
container.signature.verify	注册表、仓库、摘要、信任策略标识符和预期签名者作用域。	验证决策、签名者身份、签名与时间戳证据、策略版本；不隐式采用首次使用信任。

能力	输入	输出与护栏
container.evidence.inventory	注册表、仓库和主体摘要。	SBOM、来源、签名、扫描、VEX 和测试引用项及媒体类型与时间戳；只读。
container.registry.cleanup.recommend	注册表清单、保留策略、受保护版本和回滚窗口。	试运行删除计划、预计存储影响、被阻止主体和证据依赖；未经批准绝不删除。
container.release.assess	主体摘要、目标环境和所需策略包。	缺失或过期证据、身份与网络差距、晋升就绪程度和修复建议。

FinOps 用例不止注册表存储成本。智能体可以识别拖慢扩容的超大镜像、不必要的跨区域拉取、过期调试工件、重复架构变体、过期证据或高成本扫描模式。但不能仅因未标记清单看似未使用就删除它：摘要可能被部署、回滚记录、签名、SBOM 或模型发布引用。感知证据的清理是图问题。

10.16 关键要点

- 容器镜像是内容寻址工件图；容器是从该图创建的运行时实例。
- 标签是发现与通道引用。摘要是签名、扫描、晋升、回滚和运行时固定使用的不可变主体。
- 构建是一项围绕源代码、材料、参数、构建器身份、隔离、机密、平台和输出声明的受治理契约。
- 注册表有云控制平面和仓库数据平面。管理权限不等于拉取或推送权限。
- OCI 引用类型让 SBOM、来源、签名、扫描结果、VEX 和测试证明围绕一个摘要形成证据图。
- 没有任何单一证据类型能够证明安全。每项声明都有作用域、发行者、时间戳、验证方法和局限。
- 漏洞管理在发布后继续进行，包括重新扫描、分类、重建、替换、运行时清单和退役。
- 只有具备显式信任策略、签名者身份、摘要作用域、时间戳和使用者侧验证，签名才有用。
- 策略门禁应位于源代码、构建、注册表、晋升、准入和运行时；一次构建后扫描不能保护完整链路。
- AI 原生发布增加模型、数据、提示词、工具、安全和评估工件，它们需要同样的身份、证据与生命周期纪律。
- 首选发布模式是只构建一次、验证一次、晋升同一摘要，并持续重新评估风险。

参考资料

1. Open Container Initiative. OCI Image Specification, Version 1.1. https://specs.opencontainers.org/image-spec/

2. Open Container Initiative. OCI Distribution Specification. https://specs.opencontainers.org/distribution-spec/

3. Open Container Initiative. OCI Image and Distribution Specs v1.1 Releases. https://opencontainers.org/posts/blog/2024-03-13-image-and-distribution-1-1/

4. Open Container Initiative. About OCI. https://opencontainers.org/about/overview/

5. SLSA. SLSA Specification v1.2. https://slsa.dev/spec/v1.2/

6. SLSA. Build Provenance. https://slsa.dev/spec/v1.2/build-provenance

7. SLSA. Build Track Basics. https://slsa.dev/spec/v1.2/build-track-basics

8. in-toto. Attestation Framework v1.0 and Statement v1. https://in-toto.io/docs/specs/ and https://in-toto.io/Statement/v1

9. SPDX. SPDX Specifications. https://spdx.dev/use/specifications/

10. SPDX. SPDX Overview. https://spdx.dev/learn/overview/

11. OWASP CycloneDX. Specification Overview. https://cyclonedx.org/specification/overview/

12. OWASP CycloneDX. Machine Learning Bill of Materials. https://cyclonedx.org/capabilities/mlbom/

13. CISA. 2025 Minimum Elements for a Software Bill of Materials. https://www.cisa.gov/resources-tools/resources/2025-minimum-elements-software-bill-materials-sbom

14. CISA. Minimum Requirements for Vulnerability Exploitability eXchange (VEX). https://www.cisa.gov/resources-tools/resources/minimum-requirements-vulnerability-exploitability-exchange-vex

15. NIST. Secure Software Development Framework (SSDF) Version 1.1, SP 800-218. https://csrc.nist.gov/pubs/sp/800/218/final

16. NIST. Secure Software Development Practices for Generative AI and Dual-Use Foundation Models, SP 800-218A. https://csrc.nist.gov/pubs/sp/800/218/a/final

17. Docker. Build attestations. https://docs.docker.com/build/metadata/attestations/

18. Docker. SBOM attestations. https://docs.docker.com/build/metadata/attestations/sbom/

19. Docker. Provenance attestations. https://docs.docker.com/build/metadata/attestations/slsa-provenance/

20. Docker. docker buildx build reference. https://docs.docker.com/reference/cli/docker/buildx/build/

21. Docker. Docker Scout SBOMs. https://docs.docker.com/scout/how-tos/view-create-sboms/

22. ORAS. Attached Artifacts and Referrers. https://oras.land/docs/concepts/reftypes

23. ORAS. oras attach. https://oras.land/docs/commands/oras_attach/

24. ORAS. oras discover. https://oras.land/docs/commands/oras_discover/

25. ORAS. oras copy and referrer promotion. https://oras.land/docs/1.2/commands/oras_cp

26. Notary Project. Notation documentation. https://notaryproject.dev/docs/

27. Sigstore. Cosign signing overview. https://docs.sigstore.dev/cosign/signing/overview/

28. Sigstore. Cosign signature verification. https://docs.sigstore.dev/cosign/verifying/verify/

29. Sigstore. in-toto attestations with Cosign. https://docs.sigstore.dev/cosign/verifying/attestation/

30. Kubernetes. Images. https://kubernetes.io/docs/concepts/containers/images/

31. Kubernetes. Security Checklist. https://kubernetes.io/docs/concepts/security/security-checklist/

32. Kubernetes. Verify Signed Kubernetes Artifacts. https://kubernetes.io/docs/tasks/administer-cluster/verify-signed-artifacts/

33. Kubernetes. Verifying Container Image Signatures Within CRI Runtimes. https://kubernetes.io/blog/2023/06/29/container-image-signature-verification/

34. Microsoft Learn. Azure Container Registry concepts. https://learn.microsoft.com/azure/container-registry/container-registry-concepts

35. Microsoft Learn. Manage OCI and supply-chain artifacts with ORAS in ACR. https://learn.microsoft.com/azure/container-registry/container-registry-manage-artifact

36. Microsoft Learn. ACR Entra permissions and built-in roles. https://learn.microsoft.com/azure/container-registry/container-registry-rbac-built-in-roles-overview

37. Microsoft Learn. ACR ABAC repository permissions. https://learn.microsoft.com/azure/container-registry/container-registry-rbac-abac-repository-permissions

38. Microsoft Learn. Overview of signing and verifying OCI artifacts. https://learn.microsoft.com/azure/container-registry/overview-sign-verify-artifacts

39. Microsoft Learn. Sign container images with Notation and Azure Key Vault. https://learn.microsoft.com/azure/container-registry/container-registry-tutorial-sign-build-push

40. Microsoft Learn. Sign artifacts with Notation and Artifact Signing. https://learn.microsoft.com/azure/container-registry/container-registry-tutorial-sign-verify-notation-artifact-signing

41. Microsoft Learn. Verify image signatures with Ratify and Azure Policy on AKS. https://learn.microsoft.com/azure/container-registry/container-registry-tutorial-verify-with-ratify-aks

42. Microsoft Learn. Azure Container Registry SKU features and limits. https://learn.microsoft.com/azure/container-registry/container-registry-skus

43. Microsoft Learn. Import container images to ACR. https://learn.microsoft.com/azure/container-registry/container-registry-import-images

44. Microsoft Learn. Azure Container Registry authentication. https://learn.microsoft.com/azure/container-registry/container-registry-authentication

45. Microsoft Learn. Microsoft Defender for Containers overview. https://learn.microsoft.com/azure/defender-for-cloud/defender-for-containers-introduction

46. Alibaba Cloud. What is Container Registry (ACR). https://www.alibabacloud.com/help/en/acr/product-overview/what-is-container-registry

47. Alibaba Cloud. OCI Image and Distribution Specification v1.1.0 in ACR. https://www.alibabacloud.com/help/en/acr/use-cases/operating-with-oci-image-and-distribution-specification-v1-1-0

48. Alibaba Cloud. Container Registry Enterprise Edition billing and features. https://www.alibabacloud.com/help/en/acr/product-overview/billing-of-container-registry-enterprise-edition-instances

49. Alibaba Cloud. Container Registry event notifications. https://www.alibabacloud.com/help/en/acr/user-guide/create-an-event-notification

50. Tencent Cloud. Tencent Container Registry product overview. https://cloud.tencent.com/document/product/1141/39278

51. Tencent Cloud. Container image security scanning. https://cloud.tencent.com/document/product/1141/48185

52. Tencent Cloud. Container image signing and verification. https://cloud.tencent.com/document/product/1141/80862

53. Tencent Cloud. TCR private network access control. https://cloud.tencent.com/document/product/1141/41838

54. Tencent Cloud. TCR cross-instance and cross-account synchronization. https://cloud.tencent.com/document/product/1141/41945

55. Volcengine. Container Registry product advantages. https://www.volcengine.com/docs/6420/78485

56. Volcengine. Container Registry application scenarios and lifecycle management. https://www.volcengine.com/docs/6420/79130

57. Volcengine. VKE cr-credential-controller for password-free private image pull. https://www.volcengine.com/docs/6460/446504

58. FinOps Foundation. FinOps Framework. https://www.finops.org/framework/

59. Cloud Native Computing Foundation. Software Supply Chain Best Practices. https://tag-security.cncf.io/community/assessments/

60. IETF. CBOR Object Signing and Encryption (COSE), RFC 9052. https://www.rfc-editor.org/rfc/rfc9052

第三篇 / 第 11 章

托管 Kubernetes 与应用定义

托管控制平面、期望状态、工作负载对象、服务网络、配置和持久状态如何构成可移植的应用契约

Kubernetes 使应用架构成为可执行的定义。托管 Kubernetes 服务在该应用控制平面之外又增加了一层云控制平面，把声明式对象连接到节点池、网络、负载均衡器、磁盘、身份、策略、遥测、升级和成本。由此形成的不是“虚拟机上的容器”，而是一个分层资源系统：应用意图在其中持续与集群状态和提供商托管的基础设施进行协调。

本章内容

- 将 Kubernetes 视为嵌套在云提供商控制平面内的应用控制平面。
- 说明托管服务如何改变责任边界，却不会消除工作负载、安全、可靠性、升级或成本方面的决策。
- 使用元数据、规约、状态、控制器和协调机制，推理应用的期望状态与观测状态。
- 通过“工作负载—服务—卷”三元组对应用定义建模，并把配置、身份、策略和证据作为横切契约。
- 根据生命周期语义而非镜像类型，在 Deployment、StatefulSet、DaemonSet、Job 和 CronJob 之间作出选择。
- 把 Pod 模板定义为涵盖镜像、命令、健康探针、资源、安全上下文、身份、存储和终止行为的运行时契约。
- 把稳定的 Service 发现与不断变化的 Pod 地址分离，并比较 Service、Ingress 与 Gateway API 的职责。
- 使用 ConfigMap、Secret、ServiceAccount、RBAC、Pod Security 和 NetworkPolicy，同时避免把配置存储误当作完整的安全边界。
- 把 PVC、PV、StorageClass、CSI、回收策略、访问模式和云卷生命周期理解为一个跨控制平面的存储契约。
- 组合调度、可用性、中断、弹性、节点池和故障域策略，而不是假定仅靠副本就能获得可靠性。

- 完成一项独立 AKS 实验：部署一个包含配置、Secret 引用、Service、Gateway API 路由和持久存储的工作负载；证明 Pod 被替换后状态仍然存在；同时记录 Kubernetes 与 Azure 两侧的证据。
- 把工程意图转译到阿里云 ACK、腾讯云 TKE 和火山引擎 VKE，并记录运维上的不等价之处。

Kubernetes 经常从命令开始介绍：创建 Deployment，用 Service 将其暴露，检查 Pod，再扩展副本。这些命令很有用，却掩盖了更深层的工程变化。Kubernetes 把应用结构和生命周期转化为 API 资源。工作负载不再只是由运维人员启动的进程，而是具有所有者、选择器、Pod 模板、滚动发布策略、状态、事件和替换语义的期望状态对象。网络终结点也不再只是一个 IP 地址，而是一个 Service 和一组不断变化的就绪后端。持久存储不再只是附加到某台机器的磁盘，而是一个存储申请：通过存储类和 CSI 驱动协调为云卷，并拥有自己的生命周期与成本。

托管 Kubernetes 服务又增加了一层。云提供商创建并运维部分集群组件，把集群与身份、网络、存储、负载均衡、策略、监控、升级和计费集成，并将集群本身作为云资源公开。随后，Kubernetes 管理集群内部的应用资源。两个控制平面有所重叠，但不可互换。删除 Kubernetes Service 可能会退役一个云负载均衡器；更改 StorageClass 可能使未来的存储申请获得另一种磁盘；更改节点池会影响调度容量，却不会改变 Deployment 规约；删除控制器背后的云资源则可能引发重新协调、故障或新的成本。

本章所基于的课程使用 Service、Workload 和 Volume 三种结构抽象来解释应用定义。本章保留该三元组，并以生产环境所需的控制平面契约扩展它：配置、身份、策略、健康、证据、拓扑和成本。本章还更新了流量入口模型。Ingress 仍然重要且部署广泛，但上游 Ingress API 已被冻结，Kubernetes 建议新的可扩展路由设计采用 Gateway API。因此，持久的经验并非某个特定控制器，而是把应用路由、实现这些路由的控制器，以及代表应用创建的云基础设施彼此分离。

核心主张。托管 Kubernetes 是多个控制平面的组合。云提供商治理集群及其基础设施依赖；Kubernetes 治理应用的期望状态；应用团队仍负责声明安全、可观测、可恢复且具备成本意识的工作负载契约。

11.1 Kubernetes 是云控制平面内部的应用控制平面

Kubernetes 集群包含控制平面和工作节点。控制平面公开 API、存储资源状态、调度 Pod，并运行控制器，使观测状态趋近于期望状态。节点运行 kubelet、通过容器运行时接口（CRI）访问的容器运行时，以及网络组件。在托管服务中，提供商通常负责控制平面基础设施，而客户根据所选服务模式定义集群配置、节点池或容量策略、访问和工作负载。

这种嵌套模型之所以重要，是因为两个平面具有不同的身份、API、证据和故障模式。云控制平面创建 AKS、ACK、TKE 或 VKE 集群资源，并治理子网、路由表、负载均衡器、云磁盘、文件共享、托管身份、监控工作区、策略分配、配额与计费。Kubernetes API 则治理 Namespace、Deployment、Service、PersistentVolumeClaim、Gateway 或 Ingress 对象、ConfigMap、Secret、ServiceAccount 以及策略对象。云控制器和 CSI/CNI 集成把选定的 Kubernetes 对象转译为提供商资源。



图 11-1 托管 Kubernetes 创建嵌套的控制平面。

表 11-1 嵌套控制平面具有不同的权限范围与证据界面。

平面	主要资源	典型权限	证据
云提供商	集群、节点池、VNet/VPC、负载均衡器、磁盘/文件服务、身份、策略、监控、账单。	提供商 API、云 IAM、IaC、订阅/账户作用域。	活动/审计日志、资源图、计费记录、提供商指标、服务健康状态。
Kubernetes	Namespace、工作负载、Service、Gateway/Ingress、ConfigMap、Secret、PVC、RBAC、策略。	Kubernetes API、ServiceAccount、用户/组、RBAC、准入控制。	对象状态与条件、Event、审计日志、发布历史、控制器指标。
应用运行时	容器、进程状态、应用终结点、缓存、本地文件、业务数据。	应用身份、协议授权、进程和文件系统权限。	应用日志、指标、追踪、健康终结点、业务事件、数据记录。

cloud-controller-manager 是连接开源控制平面与提供商 API 的架构枢纽。它的控制器可以更新 Node 信息、创建路由，并为 Service 预配云负载均衡器。CSI 驱动对存储执行类似的转译。这种分离让 Kubernetes API 能表达可移植的意图，同时允许提供商演进自身的基础设施实现。可移植性存在于 API 与意图层面，并不承诺每家提供商都会创建完全相同的资源或公开完全相同的操作。

因此，一个有用的诊断问题是：哪个控制器拥有这个对象？如果 LoadBalancer Service 没有外部地址，就应检查 Service 及其 Event、云控制器集成、云权限、子网容量、配额和提供商负载均衡器状态。如果 PVC 一直处于 Pending，则应检查存储申请、StorageClass、CSI 控制器、可用区约束、配额和云存储 API。错误可能显现在 Kubernetes 中，即使被阻塞的操作实际发生在云控制平面。

11.2 托管 Kubernetes 改变责任边界

托管 Kubernetes 并非一种固定的运维模型。提供商会提供不同模式，对控制平面、系统节点池、升级、网络默认值、附加组件和容量给予不同程度的控制。Azure 区分 AKS Automatic 与 AKS Standard。阿里云区分 ACK 的不同托管版本，并且过去还提供专有控制平面模式。腾讯云提供普通集群与 Serverless 集群形式。火山引擎 VKE 同样托管核心集群组件，同时公开节点池和提供商集成。确切产品名称会演进，但责任问题保持稳定。

表 11-2 托管 Kubernetes 转移部分平台责任，但仍保留客户义务。

责任	提供商通常负责	客户仍需决定或验证
控制平面基础设施	API 服务器和状态存储托管、组件可用性、补丁、部分扩缩容与备份。	受支持版本、升级时机或通道、API 兼容性、访问策略、工作负载对控制平面可用性的依赖。
工作节点容量	节点镜像、托管节点池操作、某些模式下的自动修复或 Serverless 容量。	实例系列、可用区、污点/标签、容量上下限、中断容忍、成本、工作负载放置。
网络集成	CNI/云控制器集成、负载均衡器预配、托管 DNS 或 Ingress 附加组件。	地址规划、公网/私网暴露、出口、NetworkPolicy、路由/防火墙要求、TLS 和应用授权。
存储集成	CSI 驱动和内置 StorageClass。	访问模式、性能、可用区行为、回收策略、备份、加密边界、数据生命周期。
身份和策略	云身份挂钩、Entra/RAM/CAM/IAM 集成、策略附加组件。	人员与工作负载角色设计、ServiceAccount、最小权限、机密处理、准入策略、例外。
应用运维	通用控制器和托管附加组件。	镜像、清单、探针、资源、发布、状态、SLO、恢复、可观测性和事件响应。

最危险的误解是“控制平面由提供商托管，因此 Kubernetes 已经完全托管”。提供商可以维持 API 服务器可用，但应用仍可能因请求超过节点容量而无法调度，因副本集中在一个可用区而不可用，因 Pod 以特权方式运行而不安全，或因残留的负载均衡器和磁盘而成本高昂。托管服务质量与应用质量是两份独立契约。

节点池是尤为重要的边界。它同时是云资源模板和 Kubernetes 调度域。其实例类型、可用区、操作系统、标签、污点、扩缩容设置、维护行为和计费模式会影响 Pod 可以运行的位置，以及升级或故障将如何影响它们。对节点池模板的更改可能只应用于新节点，现有节点则可能需要轮换、重新映像、排空或升级。因此，应用团队必须为节点替换做好设计，而不能把节点当作持久服务器。

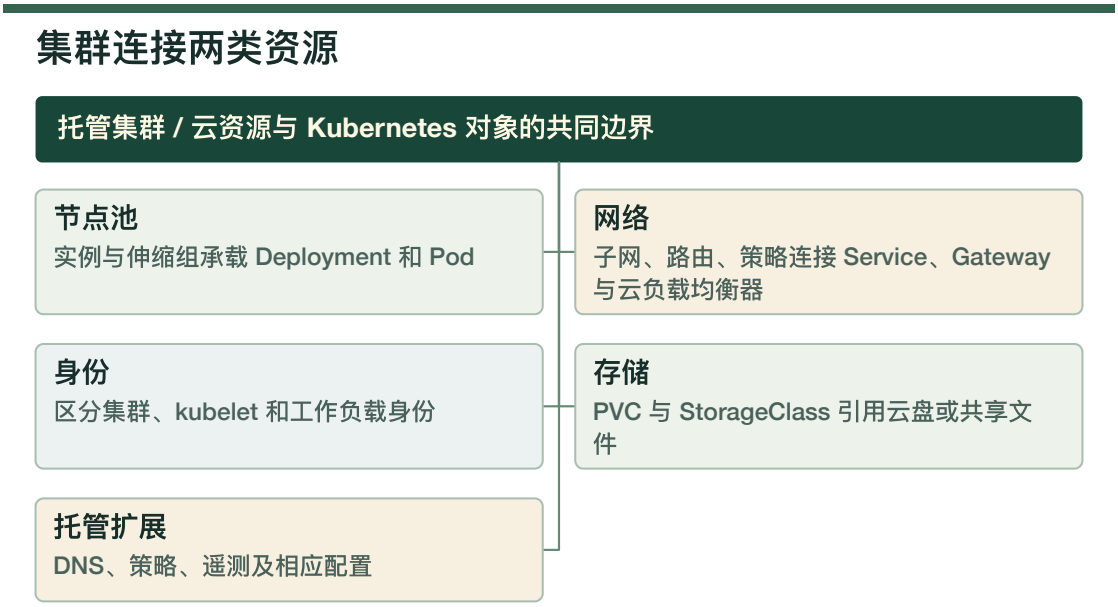


图 11-2 托管集群是提供商资源与 Kubernetes 资源构成的图。

托管模式和默认值容易变化。请在架构记录中写明集群模式、Kubernetes 版本、网络插件、Ingress 或 Gateway 实现、存储类、升级策略、支持层级和已启用的附加组件。不要仅根据提供商品牌推断这些内容。

11.3 期望状态、对象与协调

Kubernetes 对象是持久的意图声明。大多数对象包含元数据、规约和状态。元数据为对象提供名称、命名空间、标签、注解、所有权关系和生命周期元数据。规约表达期望状态，状态则记录系统报告的观测状态。控制平面持续工作，以缩小二者之间的差异。

表 11-3 对象契约把意图、身份与观测分离。

对象要素	用途	运维问题
apiVersion / kind	选择 API 组、版本和资源架构。	该 API 是否正在提供、受到支持、已弃用或已经迁移？哪个控制器理解它？
metadata	命名并分类对象，同时连接所有权或策略。	属于哪个命名空间和所有者？哪些标签用于选择？哪些注解由提供商定义？

对象要素	用途	运维问题
spec	声明期望状态。	控制器应维持什么结果？哪些字段可变、更改时需替换，或被此实现忽略？
status	报告观测状态和控制器的解释。	对象是否已 Ready、正在推进、已降级、已绑定、已准入或正在等待？控制器已经观测到哪个代次？
conditions / Events	解释状态转换和近期控制器决策。	进度为何受阻？该消息是持久证据，还是短期诊断信号？

当 Deployment 请求三个副本时，Deployment 控制器管理 ReplicaSet，由后者根据内嵌的 Pod 模板创建 Pod。调度器为未调度的 Pod 选择节点。kubelet 请求兼容 CRI 的运行时启动容器并报告状态。就绪状态决定 Pod 地址能否成为合格后端。Service 控制器或 Gateway 实现可能促使云网络资源出现。这些操作都不是一次性脚本。每个控制器都监视状态，并在故障、删除、扩缩容、发布或策略变更后再次协调。

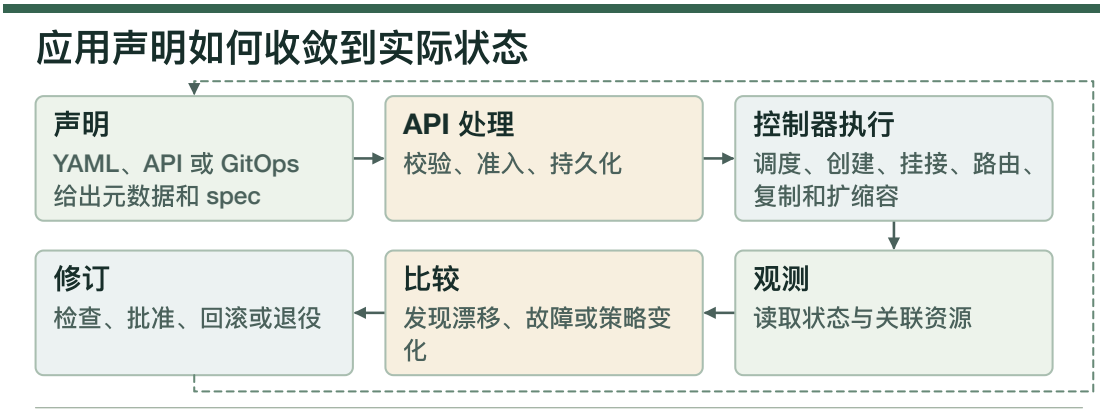


图 11-3 Kubernetes 应用定义是一份协调契约。

标签与选择器是 API 的关系纽带。Service 通过标签选择 Pod；Deployment 选择器必须匹配其 Pod 模板标签；HorizontalPodAutoscaler 以可扩缩容的工作负载为目标；策略选择命名空间、Pod 或资源。由于标签建立这些关联，随意更改标签可能使流量、指标、策略或成本归属脱离。标签属于应用架构，而不是装饰。

所有者引用与垃圾回收表达生命周期依赖。由 Deployment 拥有的 ReplicaSet 以及由 ReplicaSet 拥有的 Pod，可以通过级联删除进行清理。PVC 与为其预配的云磁盘等其他关系，则依赖回收策略，而不是简单所有权。终结器可能延迟对象删除，直到控制器完成外部清理。资源卡在 Terminating 状态，往往意味着尚未解决的外部生命周期义务，而不是删除命令失败。

声明式管理还需要字段所有权。kubectl、GitOps 控制器、Operator、准入系统和平台附加组件等多个参与者都可能修改同一个对象。服务器端应用会记录字段管理者，从而检测冲突的所有权。设计规则很简单：确定哪个控制器拥有每个字段。否则，两个协调器可能反复覆盖彼此，而表面上看二者都在正常工作。

11.4 应用定义契约

有用的应用定义始于三个结构问题：应用如何运行、如何被访问，以及状态如何延续。它们映射到工作负载、服务和卷。生产定义还要加入配置、身份、策略、拓扑、证据和经济性。应用是这些对象与关系构成的图，而不是一个 YAML 文件，更不是一个 Pod。



图 11-4 工作负载、服务和卷是应用定义的结构支柱。

表 11-4 应用定义图把每项关注点分配给一个资源系列。

关注点	主要 Kubernetes 对象	云或外部依赖
运行时生命周期	Deployment、StatefulSet、DaemonSet、Job、CronJob、Pod 模板。	节点池、容器运行时、注册表、镜像证据。
发现与流量	Service、EndpointSlice、Gateway/HTTPRoute 或 Ingress、NetworkPolicy。	负载均衡器、公网/私网 IP、DNS、防火墙/WAF、证书。
配置	ConfigMap、环境变量引用、投射卷。	配置服务、发布系统、Git 存储库。
敏感材料	Secret 引用、ServiceAccount、CSI Secret 提供程序。	云机密管理器、工作负载身份、KMS、私有终结点。
持久状态	PVC、PV、StorageClass、StatefulSet 存储申请模板、VolumeSnapshot。	云磁盘/文件/对象服务、可用区、加密、备份、成本。

关注点	主要 Kubernetes 对象	云或外部依赖
可靠性与弹性	探针、副本、PDB、HPA、拓扑约束、优先级。	节点池容量、可用区、指标流水线、提供商自动扩缩容。
治理与证据	RBAC、Pod Security、准入策略、状态、Event、审计。	云策略、监控、活动日志、计费与资源清单。

可移植性取决于提供商特定假设放在何处。Deployment 和 Service API 是可移植的，但用于请求特定负载均衡器 SKU、健康探针、内部地址或 WAF 的注解并不可移植。PVC 作为存储申请的形状可以移植，但 StorageClass 名称、参数、可用区、性能层级和快照能力各有差异。因此，可移植的应用契约应包含稳定核心以及明确的提供商覆盖层。把覆盖层隐藏在未记录的注解中并不会消除依赖，只会让依赖更难审查。

应用定义还需要版本边界。容器摘要标识运行时字节，ConfigMap 与 Secret 版本定义外部化设置，工作负载代次标识期望状态的变更。Gateway 路由、数据库架构和持久数据各有独立的迁移生命周期。只有当发布记录说明这些资源的哪些版本应协同运行时，“应用第 3 版”才有意义。

11.5 工作负载控制器编码不同的生命周期

Kubernetes 并不以一种通用类型直接管理“应用”。它提供具有不同身份、替换、顺序、完成和节点本地性语义的工作负载控制器。选错控制器会带来无论多少 YAML 模板都无法弥补的运维变通做法。



图 11-5 依据生命周期语义选择工作负载控制器。

表 11-5 工作负载控制器表达不同的替换与身份契约。

控制器	适用场景	关键语义	常见误用
Deployment	副本可互换，且应用状态位于外部。	ReplicaSet 所有权、滚动更新、回滚历史、水平扩缩容。	运行依赖身份的数据库，同时假定 Pod 名称和磁盘稳定。
StatefulSet	每个副本都需要稳定身份、有序生命周期或按副本分配的存储。	序号身份、配合 Headless Service 的稳定 DNS、有序或并行发布、volumeClaimTemplates。	假定 StatefulSet 会自动提供数据库复制、备份或正确的仲裁。
DaemonSet	某项设施必须在每个节点或选定节点上运行。	每个合格节点一个 Pod；随节点增减而变化。	把它用于普通副本，或忘记节点本地代理会占用每个池的容量。
Job	工作必须成功完成后停止。	完成数、并行度、重试、截止时间、失败策略。	把 Deployment 用于批处理，并把重启误解为任务完成。
CronJob	需要按计划创建 Job。	计划、并发策略、错过执行的行为、Job 历史。	在没有幂等任务设计的情况下假定恰好执行一次。

Pod 是最小调度单位，而不是持久的应用身份。同一 Pod 中的容器共享网络和选定的命名空间，也可以共享卷。因此，Pod 适合紧密耦合的容器，例如应用及其本地代理或初始化容器；但不适合应当独立扩缩容与失效的无关服务。工作负载控制器根据 Pod 模板创建 Pod，并在需要时予以替换。直接创建长期运行的独立 Pod，会绕过大部分生命周期契约，并使排空、修复和发布复杂化。

StatefulSet 的边界需要精确定义。它提供稳定名称和持久卷关联，但不会自动让应用成为分布式系统意义上正确的有状态应用。复制、领导者选举、一致性、备份、恢复、架构迁移和脑裂处理仍然属于应用或 Operator 的责任。同样的区别也适用于 AI 工作负载：稳定的 Pod 和磁盘本身无法构成可靠的模型服务或向量索引集群。

11.6 Pod 模板是运行时、健康与资源契约

工作负载对象中的 Pod 模板定义每个替换 Pod 应成为何种状态。由于控制器可能在不同节点重建 Pod，模板必须包含安全启动所需的一切，而不能依赖主机本地漂移。容器镜像只是该契约中的一个字段。

表 11-6 生产级 Pod 模板应明确作出以下决策。

契约领域	代表性字段或资源	重要原因
制品	使用不可变摘要指定镜像、拉取策略、需要时使用 imagePullSecrets。	把运行时绑定到经过审查的软件供应链主体。
启动	command、args、初始化容器、依赖、startupProbe。	控制初始化顺序，并区分启动缓慢与故障。
流量就绪	readinessProbe、端口、优雅终止、有充分理由时使用 preStop。	防止流量到达尚未就绪或正在终止的实例。
存活	使用范围严格受限的故障测试配置 livenessProbe。	重启卡死进程；不当的探针可能放大事件。
资源	CPU、内存和临时存储的请求与限制。	驱动调度、QoS、驱逐行为、HPA 解释和成本。
安全	runAsNonRoot、seccomp、只读根文件系统、移除的能力、禁止提权。	降低运行时权限与影响范围。
身份	ServiceAccount、令牌自动挂载决策、工作负载身份注解/标签。	定义用于访问 Kubernetes 和云资源的主体。
状态	卷、挂载、PVC 引用、emptyDir、投射配置。	明确区分临时状态与持久状态。
终止	terminationGracePeriodSeconds、信号处理、关闭证据。	允许终结点、队列和状态在替换前完成排空。

启动、就绪和存活探针回答不同问题。启动探针保护启动缓慢的应用不被存活检查过早判定失败；就绪探针询问该实例是否应接收流量；存活探针询问进程是否应重启。只要进程能够响应 HTTP 就返回成功的健康终结点，或许适合作为存活检查，却不足以判断就绪。相反，如果下游依赖不可用就让存活探针失败，则可能在依赖故障期间重启所有健康副本。

资源请求既是调度声明，也是经济声明。调度器使用请求判断哪些节点可行；限制则根据资源类型和实现约束运行时消耗。HPA 的 CPU 利用率通常以实际用量相对于请求的比例计算，因此缺失或不现实的请求会造成错误的弹性决策。QoS 类别影响压力下的驱逐行为。正确的请求值不是观测到的最大值，也不是节点容量的任意比例，而是能够支持应用 SLO 和既定突发策略的实测预留量。

安全上下文字段无法替代镜像信任或集群策略，但能让工作负载意图接受审查。命名空间级 Pod Security 策略可以拒绝不安全字段，准入策略则可以要求资源限制、获准注册表、标签或拓扑设置。第十章治理部署前的制品；本章治理制品获准如何运行。

11.7 服务发现与外部流量

Pod 可被替换，其 IP 地址也可能变化。Service 为不断变化的后端集合提供稳定名称、虚拟地址和端口契约。选择器把 Service 与 Pod 关联，EndpointSlice 则大规模表示就绪的网络终结点。这种间接层让客户端依赖服务身份，而不是 Pod 身份。

表 11-7 Service 与入口资源解决不同的流量问题。

资源	主要用途	云交互
ClusterIP Service	为一组后端提供稳定的集群内发现和虚拟 IP。	通常由集群网络和服务代理实现。
Headless Service	不使用集群虚拟 IP 的 DNS 发现，常用于 StatefulSet 身份。	依赖集群 DNS；客户端获得后端地址。
NodePort Service	在每个节点上暴露一个端口。	通常作为外部负载均衡之下的实现层；很少单独成为首选公网契约。
LoadBalancer Service	请求外部或内部四层负载均衡器。	云控制器预配提供商负载均衡器、IP、探针及相关规则。
Ingress	由 Ingress 控制器解释的传统/已冻结 Kubernetes HTTP 路由对象。	控制器可能创建或配置七层代理、负载均衡器、DNS、证书或 WAF。
Gateway API	面向角色、可扩展的四层/七层流量监听器与路由。	GatewayClass 实现负责预配并运维基础设施；路由附加受策略控制。

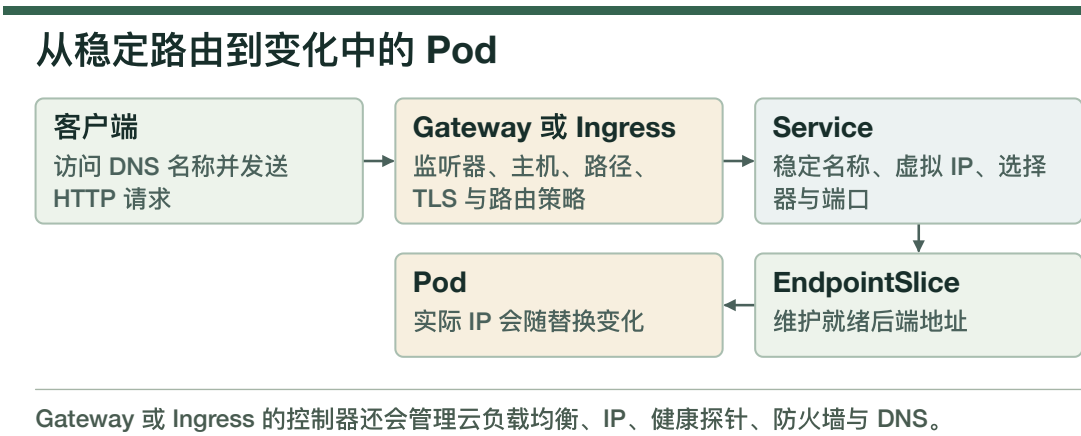


图 11-6 服务网络把稳定的路由意图与不断变化的 Pod 后端分离。

Ingress 仍然重要，因为许多集群、工具和应用都在使用它，但上游 API 已被冻结。Gateway API 通过 GatewayClass、Gateway 和 Route 资源，更明确地分离基础设施所有权与应用路由。基础设施团队可以治理监听器和共享网关，应用团队则在获准边界内附加路由。与不断增加控制器专用 Ingress 注解相比，这种面向角色的模型更适合平台工程。

路由对象本身不会承载流量，必须由控制器实现。一个已被接受的 HTTPRoute 仍可能不可用，原因包括 Gateway 没有地址、云负载均衡器尚在等待、Service 没有就绪终结点、DNS 错误、NetworkPolicy 阻塞流量、TLS 材料不可用，或应用拒绝调用者。应检查每一层的状态条件，而不是把整条路径简单归结为“Ingress 坏了”。

NetworkPolicy 只在网络实现实际强制执行策略时，才控制允许的 Pod 流量。Service 或 Gateway 不会授权业务操作。云负载均衡器安全、TLS、用户身份验证、API 授权和服务间身份仍是相互独立的控制。第七章建立的可达路径，最终仍必须落到获授权的应用操作上。

11.8 配置、机密、身份与策略

ConfigMap 和 Secret 对象都把数据从镜像中外部化，但预期敏感度不同。ConfigMap 存储非机密配置；Secret 在 Kubernetes API 中存储少量机密值，并可投射到 Pod 或由环境变量引用。Base64 编码不是加密，Kubernetes Secret 也不会自动等同于具备 HSM 支持密钥、独立访问策略、轮换、私有终结点和详细审计的云机密管理器。

表 11-8 配置与身份资源不应被当作可互换的键值存储。

机制	用途	重要边界
ConfigMap	非机密的环境特定配置或挂载文件。	更改不一定自动重启工作负载；大小和更新语义很重要。
Kubernetes Secret	由 Pod 或控制器使用的少量敏感值。	保护 API 访问和静态加密；避免宽泛 RBAC 与意外日志记录。
外部机密管理器 + **CSI/Operator**	权威的机密生命周期、轮换、审计、KMS 和私有访问。	集成身份与同步行为成为契约的一部分。
ServiceAccount	向 API 或联合层呈现的 Kubernetes 工作负载身份。	采用最小权限；当 Pod 不调用 API 时禁用令牌自动挂载。
云工作负载身份	把 ServiceAccount 联合到 Azure/RAM/CAM/IAM 主体。	云角色作用域独立于 Kubernetes RBAC 作用域。
准入与 **Pod Security** 策略	拒绝不安全或不合规的期望状态。	策略无法让已遭入侵的工作负载变得可信；需要证据与例外管理。

身份链通常包括提交清单的人员或交付系统、Pod 使用的 Kubernetes ServiceAccount、访问提供商服务的联合云身份，以及业务 API 使用的应用级身份。仅仅因为 Pod 需要读取一个 Secret 或执行一次云读取操作，就赋予其 cluster-admin 权限，会使这些边界坍塌。第五章给出了最小权限模型；其 Kubernetes 实现是用于 API 访问的 Role 或 ClusterRole 加绑定，以及用于提供商访问、单独限定作用域的云角色。

Namespace 是有用的治理边界，但并非完整的安全边界。Node、StorageClass、PersistentVolume 和某些策略对象等许多资源都属于集群作用域。除非有额外控制强制隔离，否则命名空间不会隔离内核或网络。对于强多租户，可能需要集群、节点池、运行时、网络、机密和策略边界。配套图书负责应用租户隔离的实现；本章只讨论托管集群和工作负载契约。

11.9 持久状态：PVC、PV、StorageClass 与 CSI

Kubernetes 把应用的存储请求与提供商卷分离。PersistentVolumeClaim 声明容量、访问模式和存储类。StorageClass 为预配策略和 CSI 驱动命名。当没有现有卷满足申请时，动态预配会创建 PersistentVolume 和提供商存储资源。Pod 挂载的是申请，而不是云磁盘标识符。

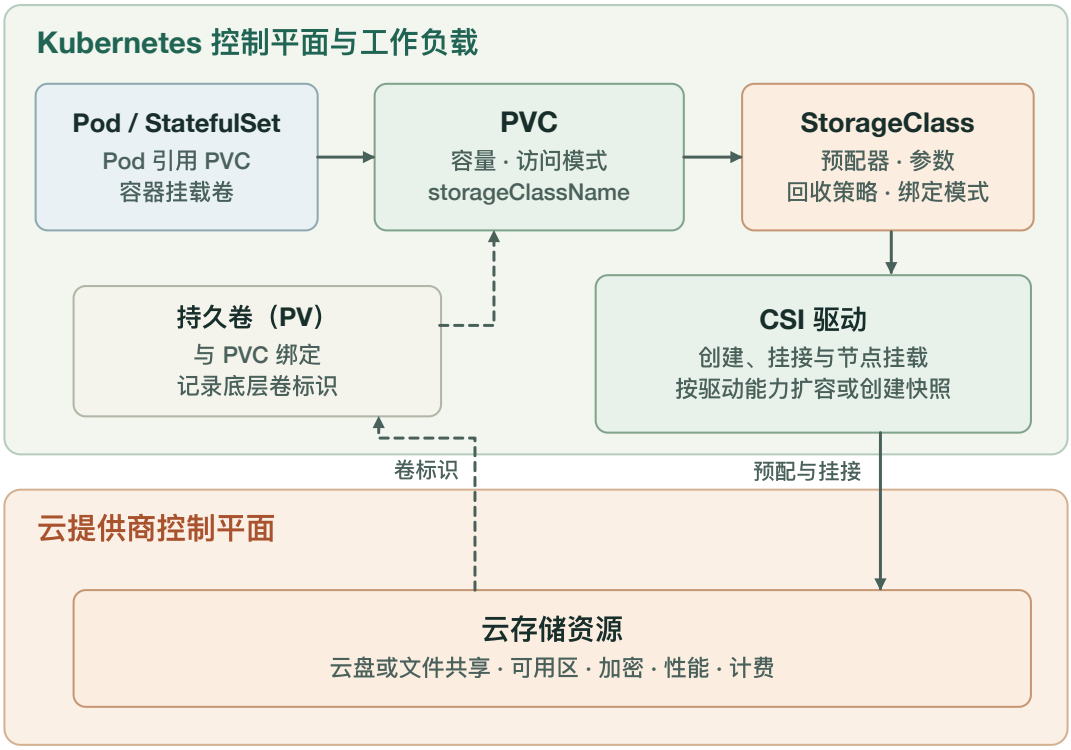


图 11-7　持久存储跨越 Kubernetes 与云控制平面。

表 11-9　持久存储对象把应用意图与基础设施实现分开。

对象或字段	责任	设计决策
PVC	应用对容量、访问模式和存储类的请求。	大小、读写共享、扩展、命名空间所有权、备份预期。

对象或字段	责任	设计决策
PV	集群对已预配或预先存在卷的表示。	绑定、容量、访问模式、回收策略、卷句柄、拓扑。
StorageClass	管理员提供的存储配置文件与预配程序参数。	性能、冗余、加密、绑定模式、扩展、回收。
CSI 驱动	用于创建、附加、挂载、调整大小、快照和生命周期操作的提供商插件。	受支持功能、身份、节点限制、版本和故障行为。
访问模式	卷可以如何由节点或 Pod 挂载。	RWO/RWX/ROX/ReadWriteOncePod 语义及后端支持。
回收策略	PV 被释放后，提供商存储会发生什么。	对一次性实验数据使用 Delete；需要独立恢复或受控处置时使用 Retain。
卷绑定模式	预配与拓扑选择发生的时机。	WaitForFirstConsumer 可以使卷的可用区与已调度 Pod 对齐。

存储生命周期独立于 Pod 生命周期。删除 Pod 通常不会删除其 PVC；当 StorageClass 使用 Delete 时，删除 PVC 可能会删除底层卷；当 PV 使用 Retain 时则会保留。StatefulSet 的 volumeClaimTemplates 按副本创建申请，但必须明确检查申请的删除行为。节点磁盘或 emptyDir 不会仅因控制器重建 Pod 就变成持久存储。

云拓扑会影响绑定。可用区磁盘只能在受支持的拓扑内附加，因此调度与预配必须在可用区上保持一致。多可用区集群不会自动让每个卷都具备多可用区能力。某些托管服务提供可用区冗余的内置存储类；另一些会创建可用区磁盘，并依靠在同一区域内重新调度，或让应用跨卷复制。架构记录必须说明真实故障域。

PVC 不是备份。快照与备份具有各自的一致性、保留、身份、恢复和跨区域契约。Pod 替换后能够成功重新挂载，只证明一个卷可以重新附加；并不能证明能够从删除、损坏、勒索软件或区域故障中恢复。

11.10 调度、可用性、中断与弹性

Kubernetes 的可用性并非由一个“高可用”开关产生。它由调度器可行性、副本放置、进程健康、服务就绪、节点池容量、中断策略、云可用区、存储拓扑和外部依赖共同构成。每一层都有不同的控制器与反应时间。



图 11-8 应用可用性是调度契约的组合。

表 11-10 可用性与弹性控制作用于不同层。

控制	它改变什么	它不能保证什么
请求/限制	调度预留、运行时约束、QoS 和驱逐行为。	节点或配额一定充足；应用一定高效使用资源。
亲和性、污点、拓扑分布	合格节点以及跨拓扑域的分布。	每个可用区都有容量，或依赖项具备可用区韧性。
副本与探针	实例数量以及流量资格或重启行为。	独立故障域、状态复制或安全关闭。
PodDisruptionBudget	通过驱逐 API 约束选定的主动驱逐。	防范节点故障、直接删除 Pod、错误发布或容量不足。
HPA / VPA	依据指标调整 Pod 副本数，或给出/分配资源建议。	立即获得节点容量，或保证指标和请求正确。
集群自动扩缩容器/节点自动预配	节点数量或节点池容量。	立即就绪；新节点需要配额、放置、引导和镜像拉取时间。
节点池可用区与升级策略	基础设施故障域和维护行为。	除非工作负载放置规则与副本数支持，否则不能保证应用分散。

PodDisruptionBudget 常被误解为可用性保证。当存在足够副本时，它约束通过驱逐 API 执行的主动驱逐；它不会防止节点故障、直接删除，或应用通过另一条路径一次替换过多实例。如果工作负载没有备用容量或选择器错误，它还可能阻塞节点维护。PDB 设计必须与发布和节点排空行为共同测试。

HPA 与节点扩缩容构成级联。HPA 可能创建更多处于 Pending 的 Pod；集群自动扩缩容器或节点自动预配层观测到不可调度的需求，并尝试增加容量。随后，提供商配额、子网地址、实例库存、可用区约束、启动时间、镜像大小和存储附加共同决定这些 Pod 何时 Ready。端到端扩缩容 SLO 应度量实际交付的服务容量，而不只是期望副本数的变化。

托管节点池可以自动执行升级、漏洞修复、修复操作或事件响应。这些操作往往会排空节点并迁移 Pod。使用本地状态、缺少 PDB、只有一个副本或无法在其他位置启动的工作负载，会把例行平台维护转化为停机。因此，应用定义必须针对托管服务预期执行的提供商操作进行设计。

11.11 托管附加组件会创建提供商资源与成本

Kubernetes 清单可以在 Kubernetes API 之外创建资源。LoadBalancer Service 或托管 Gateway 可以创建云负载均衡器、前端 IP、健康探针、规则和网络接口；PVC 可以创建磁盘或文件共享；托管 Ingress 附加组件会创建控制器和云集成；监控附加组件会创建或连接工作区与数据收集规则；工作负载身份配置则会创建联合凭据和角色分配。这些对象具有不同的所有者和删除语义。

表 11-11 Kubernetes 意图可以具体化为提供商托管的依赖资源。

Kubernetes 声明	潜在提供商资源	生命周期与成本问题
LoadBalancer Service / Gateway	负载均衡器、公网或私网 IP、健康探针、后端池、防火墙规则。	由谁删除？终结点是否公开？适用哪个 SKU、数据处理费和 IP 费用？
PVC + StorageClass	托管磁盘、文件共享、存储账户、快照资源。	Delete 还是 Retain？使用哪个可用区、冗余、性能和备份成本？
节点池	虚拟机规模集或实例组、磁盘、NIC、自动扩缩容资源。	哪些节点计费？如何处理升级、修复、缩容和 Spot 驱逐？
托管监控	日志工作区、Prometheus 工作区、代理、规则、仪表板。	收集、保留哪些信号并如何计费？是否限制了高基数数据？
工作负载身份/机密集成	托管身份、联合凭据、角色分配、Key Vault/KMS 访问。	哪个主体可以访问哪个对象，如何验证撤销？
策略与安全附加组件	策略分配、准入控制器、安全计划和发现项。	覆盖哪些资源，强制执行什么，订阅成本是多少？

控制器所有权会改变必须采用的清理方式。如果提供商负载均衡器由 Kubernetes Service 创建，直接在云控制台中删除它可能造成漂移，或导致控制器重新创建它。腾讯云明确警告：如果在拥有这些资源的 Kubernetes 工作流之外删除底层 CLB 或 CBS 资源，声明式 TKE 资源可能重新创建它们。通用规则适用于所有提供商：先退役期望状态对象，观察控制器清理，再验证提供商资源和计费作用域。

这也是成本归属应包含 Kubernetes 元数据及云资源关联的原因。Namespace、工作负载、Service、PVC、节点池和集群标签都是有用的分摊维度，但云账单可能按提供商资源 ID 列示负载均衡器、磁盘、IP、网络传输、监控摄取和工作节点实例。可靠的 FinOps 流水线在建议删除前，应先把这些提供商 ID 映射回 Kubernetes 所有者与期望状态。

11.12 运维与证据

Kubernetes 状态是分布式的。没有哪个单一界面能够证明应用健康。Deployment 可以处于 Available，而外部路由尚未解析；Gateway 可以处于 Programmed，而其 Service 没有就绪终结点；PVC 可以处于 Bound，而文件系统已满或应用无法读取。证据必须沿应用图收集。

表 11-12 应用证据包应连接期望状态、观测状态与提供商依赖。

证据	它证明什么	局限
已应用清单与摘要	已声明资源、版本、选择器、路由、存储和策略意图。	不能证明控制器已接受或实现这些内容。
对象状态与条件	控制器解释和已观测代次。	可能陈旧、因实现而异，或不足以证明最终用户行为。
Event 与发布历史	近期调度、拉取、挂载、探针、路由与发布决策。	Event 通常短暂存在，并非持久审计跟踪。
终结点测试	客户端可以到达预期路由并获得应用响应。	不能证明所有副本、身份、可用区或故障路径。
持久性替换测试	Pod 替换及卷重新附加后，状态仍然存在。	不能证明备份、损坏恢复或区域韧性。
云资源清单	已具体化哪些负载均衡器、IP、磁盘、节点组和身份。	必须与拥有它们的 Kubernetes 对象关联。
指标、日志与追踪	运行时行为、延迟、错误、饱和度与因果路径。	必须了解收集缺口、采样与保留。
清理证据	期望状态对象和提供商依赖已经退役。	软删除、保留卷、快照或延迟计费仍可能存在。

集群运维同样需要证据。记录 Kubernetes 版本、托管服务模式、附加组件版本、API 弃用发现、节点镜像、升级通道、维护窗口和支持策略。托管服务可能以不同节奏更新控制平面组件、系统附加组件或节点镜像。应用清单必须避开已移除 API，并应在集群升级前针对目标版本进行测试。

运维单位是应用资源图。kubectl get pods 必不可少，但远远不够。应结合使用发布状态、kubectl describe、对象 YAML、条件、Event、终结点测试、存储验证和云资源查询。第十三章把这些证据扩展为由 SLO 驱动的反馈回路；第十二章则会把应用变更移入受治理的 GitOps 工作流。

11.13 按工程意图进行多云转译

Kubernetes API 为多云提供了强大的共同词汇，但托管集群运维仍然因提供商而异。正确的转译单位是工程意图：托管控制平面、节点池生命周期、网络集成、存储预配、路由实现、身份联合、证据和成本。等价的 YAML 不意味着等价的故障域、附加组件成熟度、默认策略或计费方式。

表 11-13 按工程意图比较托管 Kubernetes。

意图	Azure AKS	阿里云 ACK	腾讯云 TKE	火山引擎 VKE
托管集群	AKS Automatic 或 Standard；由提供商运维控制平面，自动化边界有所不同。	ACK 托管 Basic/Pro 版；由提供商托管控制平面，并提供多种专用集群形态。	基于原生 Kubernetes 的 TKE 普通集群与 Serverless 集群。	VKE 托管集群，集成计算、网络、存储和负载均衡。
节点容量	系统/用户节点池、自动扩缩容和节点自动预配选项。	具备托管运维、扩缩容、修复、标签/污点和成本洞察的节点池。	具备生命周期管理的节点池，并可在 Pod 无法调度时自动扩容。	托管节点池与异构计算选项。
网络入口	Azure CNI 选项；LoadBalancer Service；使用 Gateway API 或其他控制器的应用路由。	Terway/flannel 选项；SLB/ALB Ingress 与云原生网关集成。	VPC-CNI 模式、CLB 支持的 Service 与 NGINX Ingress 部署模式。	VPC 集成、CLB/ALB 和 APIG Ingress 选项。
持久存储	Azure Disk、Files 与 Blob CSI；内置 managed-csi 存储类。	阿里云磁盘/NAS/OSS CSI 配置和 StorageClass。	CBS/CFS 及相关 CSI 集成。	通过 VKE 存储附加组件提供 EBS/NAS/对象集成。
身份与策略	Microsoft Entra 访问、Azure RBAC 选项、工作负载身份、Azure Policy。	RAM 与集群 RBAC、RRSA/工作负载身份模式、策略和安全集成。	CAM 加 Kubernetes RBAC；提供商身份与集群访问控制。	IAM 加 Kubernetes RBAC 和提供商集成附加组件。
证据与经济性	Azure Monitor、托管 Prometheus、活动日志、节点资源组与 Cost Management。	Prometheus/SLS/控制平面日志、节点池历史与成本洞察。	Cloud Monitor/CLS、TKE 事件与提供商计费资源。	托管可观测性、集群事件、提供商资源清单与计费。

产品版本、附加组件、受支持 Kubernetes 版本、Ingress 默认值、CNI 模式、存储类和区域功能经常变化。本表映射工程意图。生产使用前，请核实当前官方文档并记录实际集群配置。

最具可移植性的一层是应用核心：工作负载控制器、Pod 模板、Service、ConfigMap、ServiceAccount、PVC、健康探针、请求、限制与标准策略资源。提供商覆盖层应隔离在明确的文件或参数中：StorageClass、GatewayClass 或 IngressClass、负载均衡器注解、

DNS 与证书集成、工作负载身份绑定、节点池选择器，以及监控附加组件。这种结构会让不等价之处显而易见，并防止提供商特定注解悄然成为所谓可移植核心的一部分。

本章有意不讨论 Serverless Kubernetes 和基于 Kubernetes 定制的 Serverless 平台。它们会改变由谁管理节点以及工作负载如何扩缩容，却不会消除应用定义概念。配套图书将深入展开这些运行时实现。

11.14 实验 11：部署托管 Kubernetes 工作负载

本实验验证一项能力：托管 Kubernetes 集群能够协调一份完整的应用定义，其中包括配置、Secret 引用、工作负载、稳定的 Service、HTTP 入口资源和持久存储；应用保持可达，且持久状态在 Pod 替换后仍然存在；Kubernetes 与 Azure 证据可以关联；清理会移除实验所创建的期望状态对象和提供商资源。

主路径使用 Kubernetes Gateway API，因为上游 Ingress API 已被冻结，AKS 正在把托管应用路由默认值迁向 Gateway API。实验包含 Ingress 比较，但不会把它应用到基于 Gateway 的集群。本实验不是生产参考架构。它使用小型、短期集群和一个应用副本，以便保持实验聚焦并限制成本。

11.14.1 目标、架构与验收标准

表 11-14 实验 11 契约。

项目	要求
能力目标	声明并验证一个托管 Kubernetes 应用图：Namespace、ConfigMap、Secret、PVC、Deployment、Service、Gateway 和 HTTPRoute。
运行时	一个以非 root 用户运行的 Python HTTP 服务，具备启动、就绪和存活探针，并明确设置资源请求/限制。
持久状态证明	一次写入 PVC 的创建标记，在 Pod 被删除并替换后保持不变。
机密证明	应用只报告所挂载的机密文件是否存在；绝不返回机密值。
流量证明	客户端通过 Gateway 地址和 Service 后端到达应用。
控制平面证明	捕获集群配置、对象清单、状态、条件、发布、Event、PVC/PV 绑定和云磁盘。
成本边界	使用沙箱订阅、小型且受支持的双节点池或现有实验集群，以及 Standard SSD 存储，并在一次会话中删除实验环境。
清理	删除命名空间并验证动态预配的卷已经移除；如果集群是为实验创建，则删除资源组并验证无任何残留。

实验 11 的应用部署与依赖图

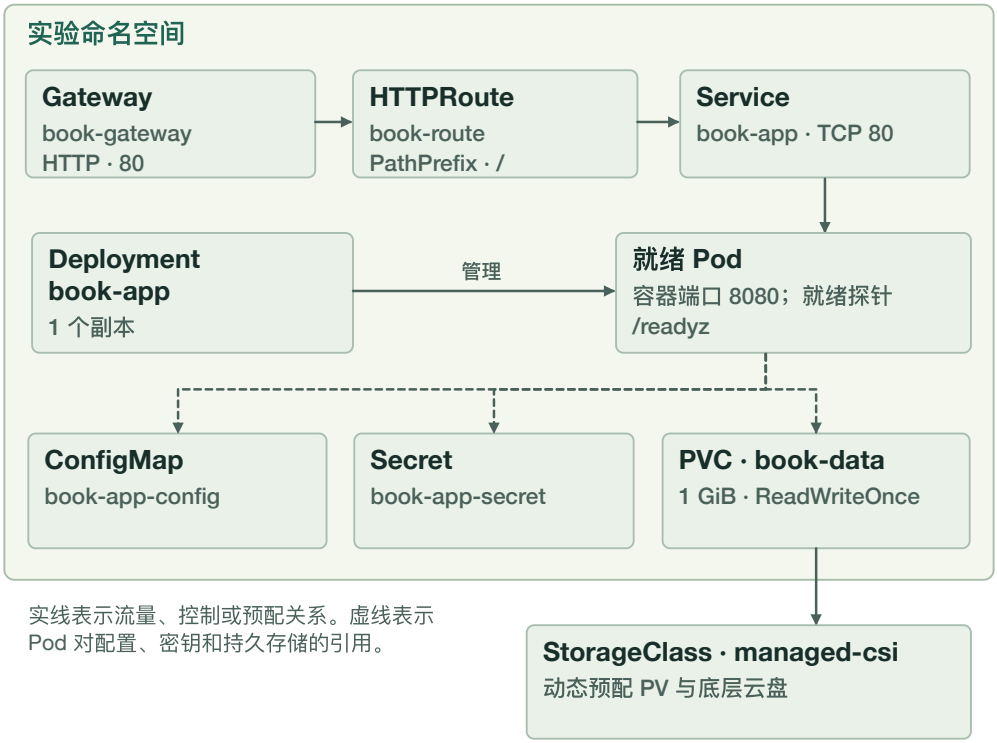


图 11-9 实验 11 部署完整的应用定义图。

11.14.2 先决条件、权限与成本边界

- 一个非生产 Azure 订阅，并有权创建资源组、AKS 集群、集成角色的集群资源、负载均衡器和托管磁盘。
- Azure CLI 2.86.0 或更高版本，用于当前 AKS 应用路由 Gateway API 命令；此外还需 kubectl、curl 和 jq。
- 所选 AKS 模式和虚拟机规格受支持的区域。如果 Standard_D2s_v5 SKU 不可用或被策略禁止，请予以替换。
- 充足的 vCPU、公网 IP、负载均衡器和托管磁盘配额。
- 具备 Bash 语义的终端。把命令版本保存在证据目录中。
- 预期成本：在集群、两个工作节点、Gateway 负载均衡器、公网 IP 和磁盘存在期间为中等。请及时删除。

如果获准的沙箱 AKS 集群已经具备托管 Gateway API 应用路由实现和 managed-csi StorageClass，请复用该集群并跳过创建集群。记录该集群为共享集群，清理时不要删除它，并使用唯一命名空间。

```
set -euo pipefail
export SUFFIX=$(openssl rand -hex 3)
export LOCATION=eastus
export RG=book-ch11-k8s-$SUFFIX
export AKS=book-aks-$SUFFIX
export NS=book-app-$SUFFIX
export EVIDENCE=$PWD/evidence-ch11-$SUFFIX
mkdir -p "$EVIDENCE"

az version > "$EVIDENCE/azure-cli-version.json"
kubectl version --client -o yaml > "$EVIDENCE/kubectl-client-version.yaml"
az group create --name "$RG" --location "$LOCATION"   --tags chapter=11
purpose=managed-kubernetes owner=book-lab
```

11.14.3 创建或复用短期 AKS 集群

以下命令创建一个双节点 AKS Standard 集群，使用 Azure CNI Overlay，启用 OIDC 颁发者和工作负载身份，以及托管 Gateway API 应用路由实现。即使示例应用不会调用 Azure 数据平面 API，实验仍有意启用工作负载身份；这样，集群配置便会展现本章讨论的身份边界，而无需在 Pod 中放置凭据。

```
az aks create   --resource-group "$RG"   --name "$AKS"   --location "$LOCATION"
--node-count 2   --node-vm-size Standard_D2s_v5   --enable-managed-identity   --
enable-oidc-issuer   --enable-workload-identity   --network-plugin azure   --
network-plugin-mode overlay   --enable-app-routing-istio   --generate-ssh-keys

az aks get-credentials --resource-group "$RG" --name "$AKS" --overwrite-existing
az aks show -g "$RG" -n "$AKS" > "$EVIDENCE/aks-resource.json"
kubectl version -o yaml > "$EVIDENCE/kubernetes-version.yaml"
kubectl get nodes -o wide > "$EVIDENCE/nodes.txt"
kubectl get gatewayclass > "$EVIDENCE/gatewayclasses.txt"
kubectl get storageclass > "$EVIDENCE/storageclasses.txt"
```

AKS 功能和 CLI 标志会持续演进。如果托管 Gateway API 选项在所选区域或租户中不可用，请使用现有的获准实现，并记录其 GatewayClass、控制器、版本、支持边界与云资源。不要在未说明的情况下替换实现。

11.14.4 创建应用定义

创建一份清单。ConfigMap 包含非机密的应用代码和环境消息。Secret 包含占位令牌。应用只验证机密文件是否已挂载。Standard SSD 存储申请使用内置 managed-csi StorageClass。初始化容器仅在卷为空时写入创建标记，从而让实验证明状态能在 Pod 替换后延续。

```
cat > book-app.yaml <<'YAML'
apiVersion: v1
kind: Namespace
metadata:
  name: REPLACE_NAMESPACE
  labels:
    pod-security.kubernetes.io/enforce: restricted
    pod-security.kubernetes.io/audit: restricted
    pod-security.kubernetes.io/warn: restricted
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: book-app-config
  namespace: REPLACE_NAMESPACE
data:
  APP_MESSAGE: "Managed Kubernetes reconciles application intent."
  app.py: |
    import json, os
    from http.server import BaseHTTPRequestHandler, ThreadingHTTPServer
    from pathlib import Path

    DATA = Path('/data')
    SECRET = Path('/var/run/book-secret/token')

    class Handler(BaseHTTPRequestHandler):
        def reply(self, status, payload):
            body = json.dumps(payload, sort_keys=True).encode()
            self.send_response(status)
            self.send_header('Content-Type', 'application/json')
            self.send_header('Content-Length', str(len(body)))
            self.end_headers()
            self.wfile.write(body)

        def do_GET(self):
            if self.path in ('/healthz', '/readyz'):
                return self.reply(200, {'status': 'ok'})
            if self.path != '/':
                return self.reply(404, {'error': 'not found'})
            created = (DATA / 'created-at.txt').read_text().strip()
            secret_state = 'mounted' if SECRET.is_file() and
SECRET.stat().st_size > 0 else 'missing'
            self.reply(200, {
                'service': 'book-managed-kubernetes',
                'message': os.getenv('APP_MESSAGE', 'unset'),
                'createdAt': created,
                'secretState': secret_state,
                'pod': os.getenv('HOSTNAME', 'unknown')
```

```
            })

        def log_message(self, fmt, *args):
            print('%s - %s' % (self.address_string(), fmt % args), flush=True)

    ThreadingHTTPServer(('0.0.0.0', 8080), Handler).serve_forever()
---
apiVersion: v1
kind: Secret
metadata:
  name: book-app-secret
  namespace: REPLACE_NAMESPACE
type: Opaque
stringData:
  token: "lab-placeholder-do-not-use-in-production"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: book-data
  namespace: REPLACE_NAMESPACE
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: managed-csi
  resources:
    requests:
      storage: 1Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: book-app
  namespace: REPLACE_NAMESPACE
  labels:
    app.kubernetes.io/name: book-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: book-app
  template:
    metadata:
      labels:
        app.kubernetes.io/name: book-app
    spec:
      automountServiceAccountToken: false
      securityContext:
```

```
        runAsNonRoot: true
        runAsUser: 10001
        runAsGroup: 10001
        fsGroup: 10001
        seccompProfile:
          type: RuntimeDefault
      initContainers:
        - name: initialize-persistent-state
          image: python:3.12-alpine
          command: ["/bin/sh", "-c"]
          args:
            - |
              set -eu
              if [ ! -f /data/created-at.txt ]; then
                date -u +%Y-%m-%dT%H:%M:%SZ > /data/created-at.txt
              fi
              test -s /var/run/book-secret/token
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop: ["ALL"]
          volumeMounts:
            - name: data
              mountPath: /data
            - name: secret
              mountPath: /var/run/book-secret
              readOnly: true
      containers:
        - name: app
          image: python:3.12-alpine
          command: ["python", "/app/app.py"]
          env:
            - name: APP_MESSAGE
              valueFrom:
                configMapKeyRef:
                  name: book-app-config
                  key: APP_MESSAGE
            - name: PYTHONDONTWRITEBYTECODE
              value: "1"
            - name: TMPDIR
              value: /tmp
          ports:
            - name: http
              containerPort: 8080
          startupProbe:
            httpGet: { path: /healthz, port: http }
            failureThreshold: 30
            periodSeconds: 2
```

```
          readinessProbe:
            httpGet: { path: /readyz, port: http }
            periodSeconds: 5
          livenessProbe:
            httpGet: { path: /healthz, port: http }
            periodSeconds: 10
          resources:
            requests: { cpu: 50m, memory: 64Mi }
            limits: { cpu: 250m, memory: 192Mi }
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            capabilities:
              drop: ["ALL"]
          volumeMounts:
            - name: app-code
              mountPath: /app
              readOnly: true
            - name: data
              mountPath: /data
            - name: secret
              mountPath: /var/run/book-secret
              readOnly: true
            - name: tmp
              mountPath: /tmp
      volumes:
        - name: app-code
          configMap:
            name: book-app-config
            items:
              - key: app.py
                path: app.py
        - name: data
          persistentVolumeClaim:
            claimName: book-data
        - name: secret
          secret:
            secretName: book-app-secret
        - name: tmp
          emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: book-app
  namespace: REPLACE_NAMESPACE
spec:
  selector:
```

```
    app.kubernetes.io/name: book-app
  ports:
    - name: http
      port: 80
      targetPort: http
---
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: book-gateway
  namespace: REPLACE_NAMESPACE
spec:
  gatewayClassName: approuting-istio
  listeners:
    - name: http
      protocol: HTTP
      port: 80
      allowedRoutes:
        namespaces:
          from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: book-route
  namespace: REPLACE_NAMESPACE
spec:
  parentRefs:
    - name: book-gateway
  rules:
    - matches:
        - path:
            type: PathPrefix
            value: /
      backendRefs:
        - name: book-app
          port: 80
YAML

sed -i "s/REPLACE_NAMESPACE/$NS/g" book-app.yaml
cp book-app.yaml "$EVIDENCE/book-app.yaml"
```

实验使用易读的 Python 基础镜像标签。生产清单应引用获准的摘要，并应从外部机密生命周期获取敏感材料，而不是在文件中放置字面量 stringData。这里的占位 Secret 仅用于演示资源连接。

11.14.5 应用并检查期望状态

```
kubectl apply -f book-app.yaml
kubectl rollout status deployment/book-app -n "$NS" --timeout=10m
kubectl wait --for=jsonpath='{.status.phase}'=Bound pvc/book-data -n "$NS" --
timeout=10m

kubectl get deployment,pod,service,pvc,gateway,httproute -n "$NS" -o wide   |
tee "$EVIDENCE/application-summary.txt"
kubectl get deployment/book-app -n "$NS" -o yaml > "$EVIDENCE/deployment.yaml"
kubectl get gateway/book-gateway -n "$NS" -o yaml > "$EVIDENCE/gateway.yaml"
kubectl get httproute/book-route -n "$NS" -o yaml > "$EVIDENCE/httproute.yaml"
kubectl get pvc/book-data -n "$NS" -o yaml > "$EVIDENCE/pvc.yaml"
kubectl get events -n "$NS" --sort-by=.lastTimestamp > "$EVIDENCE/events-before-
test.txt"
```

检查 status.observedGeneration、Deployment 条件、Pod 就绪状态、PVC 阶段、Gateway 条件和 HTTPRoute 父级状态。应用成功只表示 API 已接受请求。验收测试还要求相关控制器报告状态已经实现。

11.14.6 验证 HTTP 入口路径

等待托管 Gateway 获得地址。该地址可能是 IP 或主机名。测试应用前保存路由状态。

```
for i in $(seq 1 90); do
  ADDRESS=$(kubectl get gateway book-gateway -n "$NS"     -o
jsonpath='{.status.addresses[0].value}' 2>/dev/null || true)
  [ -n "$ADDRESS" ] && break
  sleep 10
done

if [ -z "${ADDRESS:-}" ]; then
  kubectl describe gateway book-gateway -n "$NS"
  kubectl describe httproute book-route -n "$NS"
  exit 1
fi

echo "$ADDRESS" | tee "$EVIDENCE/gateway-address.txt"
curl --retry 20 --retry-delay 5 --fail --show-error   "http://$ADDRESS/" | tee
"$EVIDENCE/response-before.json"
jq . "$EVIDENCE/response-before.json"
```

预期响应包含已配置消息、持久的 createdAt 时间戳、secretState: mounted 以及当前 Pod 名称。响应不得包含 Secret 值。如果 Gateway 已有地址而 curl 失败，请先检查 HTTPRoute 的 Accepted 和 ResolvedRefs 条件、Service 终结点、Pod 就绪状态及 Gateway 实现日志，再更改应用。

11.14.7 证明持久状态能在 Pod 替换后延续

```
BEFORE=$(jq -r .createdAt "$EVIDENCE/response-before.json")
OLD_POD=$(kubectl get pod -n "$NS" -l app.kubernetes.io/name=book-app   -o
jsonpath='{.items[0].metadata.name}')

echo "$OLD_POD" | tee "$EVIDENCE/old-pod.txt"
kubectl delete pod "$OLD_POD" -n "$NS"
kubectl rollout status deployment/book-app -n "$NS" --timeout=10m

NEW_POD=$(kubectl get pod -n "$NS" -l app.kubernetes.io/name=book-app   -o
jsonpath='{.items[0].metadata.name}')
echo "$NEW_POD" | tee "$EVIDENCE/new-pod.txt"

curl --retry 20 --retry-delay 5 --fail --show-error   "http://$ADDRESS/" | tee
"$EVIDENCE/response-after.json"
AFTER=$(jq -r .createdAt "$EVIDENCE/response-after.json")

test "$OLD_POD" != "$NEW_POD"
test "$BEFORE" = "$AFTER"
printf 'persistent timestamp: %s
' "$AFTER"   | tee "$EVIDENCE/persistence-verification.txt"
```

该测试证明的主张范围很窄：替换 Pod 挂载了同一个持久卷，并观测到原始标记。它不能证明多可用区可用性、备份、应用一致性恢复或防止数据损坏。记录主张时应严格限定在实际测试的范围。

11.14.8 检查 Kubernetes 创建的云资源

```
PV=$(kubectl get pvc book-data -n "$NS" -o jsonpath='{.spec.volumeName}')
VOLUME_ID=$(kubectl get pv "$PV" -o jsonpath='{.spec.csi.volumeHandle}')
NODE_RG=$(az aks show -g "$RG" -n "$AKS" --query nodeResourceGroup -o tsv)

printf 'PV=%s
VOLUME_ID=%s
NODE_RG=%s
' "$PV" "$VOLUME_ID" "$NODE_RG"   | tee "$EVIDENCE/cloud-links.txt"
kubectl get pv "$PV" -o yaml > "$EVIDENCE/pv.yaml"
az disk show --ids "$VOLUME_ID" > "$EVIDENCE/azure-disk.json"
az resource list -g "$NODE_RG" > "$EVIDENCE/node-resource-group.json"
az network lb list -g "$NODE_RG" > "$EVIDENCE/load-balancers.json"
az network public-ip list -g "$NODE_RG" > "$EVIDENCE/public-ips.json"
```

证据应把 PVC 连接到其 PV 与 CSI volumeHandle，再连接到 Azure 托管磁盘；还应显示托管 Gateway 实现使用的负载均衡资源。名称与内部拓扑因实现而异，应使用所有者链接、资源 ID、标签与状态，而不能只依赖命名模式。

11.14.9 构建证据包

表 11-15 实验 11 所需证据。

证据文件	用途
aks-resource.json / kubernetes-version.yaml	记录托管服务模式、身份、网络配置、OIDC 设置和 Kubernetes 版本。
book-app.yaml	记录期望应用状态以及提供商特定的 GatewayClass/StorageClass 选择。
deployment.yaml / gateway.yaml / httproute.yaml	记录已观测代次、条件和控制器状态。
pvc.yaml / pv.yaml / azure-disk.json	把应用存储意图连接到提供商卷。
response-before.json / response-after.json	证明路由行为、配置注入、机密在场但未泄露，以及状态连续性。
old-pod.txt / new-pod.txt / persistence-verification.txt	证明替换已经发生，且标记保持不变。
events-before-test.txt / events-after-test.txt	捕获调度、拉取、挂载、探针与路由诊断。
node-resource-group.json / load-balancers.json / public-ips.json	显示为实现 Kubernetes 意图而创建的提供商资源。
清理文件	证明命名空间、磁盘以及（若为实验创建）集群资源组已经退役。

```
kubectl get events -n "$NS" --sort-by=.lastTimestamp   > "$EVIDENCE/events-
after-test.txt"
kubectl auth can-i --list -n "$NS" > "$EVIDENCE/current-user-k8s-
permissions.txt"
sha256sum "$EVIDENCE"/* > "$EVIDENCE/evidence-packet.sha256"
ls -lh "$EVIDENCE"
```

11.14.10 Ingress 兼容性说明

对于有意使用 AKS 托管 NGINX 应用路由附加组件的集群，等价的传统入口对象将使用 ingressClassName: webapprouting.kubernetes.azure.com。除非安装了 NGINX 控制器，否则不要把该对象应用到仅使用 Gateway 的实验集群。保持 Service 不变，并用以下资源替换 Gateway/HTTPRoute 组合：

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: book-ingress
  namespace: REPLACE_NAMESPACE
spec:
  ingressClassName: webapprouting.kubernetes.azure.com
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: book-app
                port:
                  number: 80
```

应用意图相同——把 HTTP 流量路由到 book-app Service——但对象模型、控制器支持生命周期、角色分离和生成的基础设施有所不同。这是接口相似但运维并不完全等价的具体示例。

11.14.11 常见故障及其含义

表 11-16 实验故障模式与诊断问题。

症状	可能所在层	诊断路径
Pod Pending	调度/容量	描述 Pod；检查请求、节点污点、配额、节点就绪状态和可用区约束。
ImagePullBackOff	注册表/出口/制品	检查镜像引用、DNS/出口、注册表可用性和拉取凭据。
PVC Pending	存储控制路径	描述 PVC；检查 StorageClass、CSI Pod、配额、拓扑和 Event。
Pod Running 但未 Ready	应用健康	检查就绪终结点、日志、配置、卷挂载和探针时机。
Gateway 没有地址	Gateway 控制器/云网络	检查 Gateway 条件、控制器命名空间、提供商配额、子网/IP 与云活动日志。
HTTPRoute 未被 Accepted	路由策略或引用	检查 parentRefs、allowedRoutes、后端 Service 名称/端口和 ResolvedRefs 条件。

症状	可能所在层	诊断路径
Curl 返回 503	Service 后端	检查 EndpointSlice 与 Pod 就绪状态；确认 Service 选择器匹配模板标签。
Multi-Attach 错误	卷访问/拓扑	确认 RWO 磁盘只使用一个副本、先前附加已释放、节点可用区及 CSI 状态。
替换后时间戳发生变化	持久挂载或初始化逻辑	确认 PVC 挂载、PV 身份、回收行为和应用路径。

11.14.12 清理与零残留验证

先删除命名空间，以便实验观测由 Kubernetes 驱动的存储申请及其提供商卷清理。删除前保存卷 ID。如果集群仅为本实验创建，随后删除资源组。如果是共享集群，则不要删除集群或节点资源组。

```
kubectl delete namespace "$NS"
kubectl wait --for=delete namespace/"$NS" --timeout=15m || true

# Verify the dynamically provisioned disk disappears under the Delete reclaim
policy.
for i in $(seq 1 60); do
  if ! az disk show --ids "$VOLUME_ID" >/dev/null 2>&1; then
    echo "disk deleted" | tee "$EVIDENCE/disk-cleanup.txt"
    break
  fi
  sleep 10
done

kubectl get namespace "$NS" > "$EVIDENCE/namespace-after-cleanup.txt" 2>&1 ||
true

# Run only when this lab created the cluster resource group.
az group delete --name "$RG" --yes --no-wait
az group wait --deleted --name "$RG"
az group exists --name "$RG" | tee "$EVIDENCE/resource-group-exists-after-
cleanup.txt"
```

使用 Retain 的自定义 StorageClass、快照、备份保管库、静态公网 IP、DNS 区域或外部证书，都可能在删除命名空间后继续存在。清理证据必须沿实际资源图收集，不能假定删除集群就会移除每项相关资源。

11.14.13 将实验转译到中国领先云平台

表 11-17 按托管 Kubernetes 能力转译实验 11。

实验意图	ACK	TKE	VKE
托管集群与节点池	ACK 托管集群和节点池；记录版本、可用区、CNI 与托管运维设置。	TKE 普通集群和节点池；记录运行时、VPC 模式、扩缩容与版本支持。	VKE 托管集群和托管节点池；记录网络、异构容量与生命周期设置。
HTTP 入口	使用受支持的 ALB/SLB Ingress 或 Gateway 实现，并记录类/控制器以及创建的云资源。	使用 CLB 支持的 Service 或获准的 NGINX/Gateway 实现，并记录控制器所有权。	使用受支持的 ALB、CLB 或 APIG Ingress/Gateway 集成，并记录所预配的入口资源。
持久存储	选择获准的磁盘或 NAS CSI StorageClass，并验证从 PV 卷句柄到提供商资源的关联。	使用 CBS 或 CFS CSI 存储类，并验证申请、PV 与提供商存储生命周期。	使用 VKE 支持的块/文件 CSI 存储类，并验证绑定、可用区与回收行为。
配置与机密引用	使用标准 ConfigMap/Secret 核心对象；在支持时把生产机密外部化到 KMS/机密服务。	使用标准核心对象，结合 CAM/RBAC 与获准的机密集成。	使用标准核心对象，结合 IAM/RBAC 与获准的机密集成。
证据与清理	捕获 ACK Event/控制平面日志、SLS/Prometheus 证据、提供商资源 ID 与成本作用域。	捕获 TKE Event、CloudAudit/CLS/监控以及 CLB/CBS 资源；通过拥有这些资源的 Kubernetes 对象执行删除。	捕获 VKE 状态、可观测性、审计和依赖云资源；验证计费清理。

11.15 CloudDesk 复用

CloudDesk FinOps Agent 不应获得不受限的 kubectl、云 CLI 或 cluster-admin 凭据。本章产生的应是有界限、主要面向读取的能力，可由工具代理程序实现，并显式限定集群、命名空间、对象种类和云账户作用域。变更操作应保持独立的“计划—批准—执行—验证”工作流。

表 11-18 CloudDesk 能力交接。

能力	读取契约	可能的受治理操作
k8s.workload.inventory	在获准作用域内列出工作负载、副本、镜像、请求/限制、探针、放置和状态。	提出资源或副本更改建议；未经批准及发布验证，不得直接修补。
k8s.ingress.audit	映射 Gateway/Ingress、Service、EndpointSlice 和提供商负载均衡器的暴露关系。	结合网络和所有权证据，提出内部化、TLS、WAF 或路由清理建议。

能力	读取契约	可能的受治理操作
k8s.storage.recommend	把 PVC/PV/StorageClass 映射到提供商卷、访问模式、回收策略、可用区、利用率和成本。	通过经过审查的计划，提出扩容、存储类迁移、快照/备份或孤立资源清理建议。
k8s.application.definition.validate	评估对象图完整性、选择器、探针、资源、身份、策略和提供商覆盖层。	创建 GitOps 变更提案，而不是变更实时对象。
k8s.drift.explain	比较期望清单、对象代次、控制器状态与提供商资源状态。	建议负责协调的所有者以及要收集的证据；避免与现有控制器竞争。

对于 CloudDesk FinOps 用例，最有价值的连接位于 Kubernetes 意图与提供商计费之间。在智能体证明哪个 Service 或 Gateway 拥有某个“未使用”负载均衡器、该资源是否仍为期望状态、哪些路由依赖它，以及删除后是否会被协调器再次创建之前，建议删除并不安全。同样，磁盘没有附加到虚拟机，也不意味着它是孤立资源；它可能绑定到 PVC，或为了恢复而保留。资源所有权先于成本操作。

11.16 要点

- Kubernetes 是应用控制平面；托管 Kubernetes 服务把它嵌套在提供商云控制平面内。
- 托管控制平面的可用性不会消除客户在工作负载定义、访问、放置、状态、SLO、升级、证据或成本方面的责任。
- Kubernetes 对象把身份和期望规约与观测状态分离；控制器持续协调二者之间的差异。
- 应用定义是一幅资源图。工作负载、Service 和卷是结构支柱，周围环绕配置、身份、策略、拓扑、证据与经济性。
- Deployment、StatefulSet、DaemonSet、Job 与 CronJob 编码不同的生命周期语义。Pod 是可替换的调度单位，而非持久应用身份。
- Pod 模板必须声明制品、启动、健康、资源、安全、身份、状态和终止行为。
- Service 与 EndpointSlice 在不断变化的 Pod 之上建立稳定发现。Ingress 已被冻结；Gateway API 提供面向角色的后继方案，但二者都需要实现控制器。
- ConfigMap、Secret、外部机密管理器、ServiceAccount、云工作负载身份和准入策略解决的是不同问题。
- PVC、PV、StorageClass 与 CSI 把应用存储意图转译为提供商卷；回收、拓扑、备份与恢复仍需明确决策。
- 副本、探针、PDB、HPA、拓扑分布和节点扩缩容作用于不同层，必须作为一条可用性路径共同测试。
- Kubernetes 对象可以创建会产生费用的云资源。清理必须从拥有它们的期望状态对象开始，以提供商资源和计费验证结束。

- AKS、ACK、TKE 与 VKE 上等价的 Kubernetes YAML 并不能确立运维等价性；应记录控制器、默认值、故障域、附加组件、证据和成本。
- 实验 11 证明了一项范围狭窄但很有价值的契约：完整应用图得到协调、可从外部访问、持久状态可跨 Pod 替换延续，并能追溯到其 Azure 资源。
- 智能体工具应使用有界限的清单与解释契约，而不是不受限的集群管理权限。

参考资料与官方文档

[1] 美国国家标准与技术研究院。《NIST 云计算定义》。NIST SP 800-145，2011。https://doi.org/10.6028/NIST.SP.800-145

[2] Kubernetes。《集群架构》。https://kubernetes.io/docs/concepts/architecture/

[3] Kubernetes。《Kubernetes 中的对象》。https://kubernetes.io/docs/concepts/overview/working-with-objects/

[4] Kubernetes。《控制器》。https://kubernetes.io/docs/concepts/architecture/controller/

[5] Kubernetes。《云控制器管理器》。https://kubernetes.io/docs/concepts/architecture/cloud-controller/

[6] Kubernetes。《节点》。https://kubernetes.io/docs/concepts/architecture/nodes/

[7] Kubernetes。《Kubernetes 自愈》。https://kubernetes.io/docs/concepts/architecture/self-healing/

[8] Kubernetes。《Pod》。https://kubernetes.io/docs/concepts/workloads/pods/

[9] Kubernetes。《Pod 生命周期》。https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/

[10] Kubernetes。《工作负载》。https://kubernetes.io/docs/concepts/workloads/

[11] Kubernetes。《Deployment》。https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

[12] Kubernetes。《StatefulSet》。https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

[13] Kubernetes。《DaemonSet》。https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

[14] Kubernetes。《Job》。https://kubernetes.io/docs/concepts/workloads/controllers/job/

[15] Kubernetes。《CronJob》。
https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

[16] Kubernetes。《命名空间》。
https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

[17] Kubernetes。《标签与选择器》。
https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

[18] Kubernetes。《垃圾回收》。
https://kubernetes.io/docs/concepts/architecture/garbage-collection/

[19] Kubernetes。《ConfigMap》。
https://kubernetes.io/docs/concepts/configuration/configmap/

[20] Kubernetes。《Secret》。
https://kubernetes.io/docs/concepts/configuration/secret/

[21] Kubernetes。《ServiceAccount》。
https://kubernetes.io/docs/concepts/security/service-accounts/

[22] Kubernetes。《基于角色的访问控制最佳实践》。
https://kubernetes.io/docs/concepts/security/rbac-good-practices/

[23] Kubernetes。《Pod 安全标准》。
https://kubernetes.io/docs/concepts/security/pod-security-standards/

[24] Kubernetes。《Pod Security 准入》。
https://kubernetes.io/docs/concepts/security/pod-security-admission/

[25] Kubernetes。《Pod 与容器的资源管理》。
https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

[26] Kubernetes。《Pod 服务质量类别》。
https://kubernetes.io/docs/concepts/workloads/pods/pod-qos/

[27] Kubernetes。《配置存活、就绪和启动探针》。
https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/

[28] Kubernetes。《Pod 水平自动扩缩容》。
https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/

[29] Kubernetes。《中断与 PodDisruptionBudget》。
https://kubernetes.io/docs/concepts/workloads/pods/disruptions/

[30] Kubernetes。《Pod 拓扑分布约束》。
https://kubernetes.io/docs/concepts/scheduling-eviction/topology-spread-constraints/

[31] Kubernetes。《Service》。https://kubernetes.io/docs/concepts/services-networking/service/

[32] Kubernetes。《EndpointSlice》。https://kubernetes.io/docs/concepts/services-networking/endpoint-slices/

[33] Kubernetes。《Ingress》。https://kubernetes.io/docs/concepts/services-networking/ingress/

[34] Kubernetes。《Gateway API》。https://kubernetes.io/docs/concepts/services-networking/gateway/

[35] Kubernetes。《NetworkPolicy》。https://kubernetes.io/docs/concepts/services-networking/network-policies/

[36] Kubernetes。《PersistentVolume》。
https://kubernetes.io/docs/concepts/storage/persistent-volumes/

[37] Kubernetes。《StorageClass》。
https://kubernetes.io/docs/concepts/storage/storage-classes/

[38] Kubernetes。《动态卷预配》。
https://kubernetes.io/docs/concepts/storage/dynamic-provisioning/

[39] Kubernetes。《卷》。https://kubernetes.io/docs/concepts/storage/volumes/

[40] Kubernetes。《卷快照》。https://kubernetes.io/docs/concepts/storage/volume-snapshots/

[41] Kubernetes。《容器运行时接口》。
https://kubernetes.io/docs/concepts/architecture/cri/

[42] Microsoft。《什么是 Azure Kubernetes 服务（AKS）？》
https://learn.microsoft.com/en-us/azure/aks/what-is-aks

[43] Microsoft。《Azure Kubernetes 服务核心概念》。更新于 2026 年 7 月 29 日。
https://learn.microsoft.com/en-us/azure/aks/core-aks-concepts

[44] Microsoft。《Azure Kubernetes 服务支持策略》。更新于 2026 年 6 月 9 日。https://learn.microsoft.com/en-us/azure/aks/support-policies

[45] Microsoft。《Azure Kubernetes 服务中的托管身份》。更新于 2026 年 8 月 18 日。https://learn.microsoft.com/en-us/azure/aks/managed-identity-overview

[46] Microsoft。《在 AKS 上部署和配置 Microsoft Entra Workload ID》。https://learn.microsoft.com/en-us/azure/aks/workload-identity-deploy-cluster

[47] Microsoft。《Azure Kubernetes 服务中的网络概念》。更新于 2026 年 4 月 13 日。https://learn.microsoft.com/en-us/azure/aks/concepts-network

[48] Microsoft。《AKS 应用与集群安全概念》。更新于 2026 年 6 月 4 日。https://learn.microsoft.com/en-us/azure/aks/concepts-security

[49] Microsoft。《AKS 应用的存储选项》。更新于 2026 年 7 月 31 日。https://learn.microsoft.com/en-us/azure/aks/concepts-storage

[50] Microsoft。《在 AKS 上使用 CSI 驱动》。更新于 2026 年 8 月 3 日。https://learn.microsoft.com/en-us/azure/aks/csi-storage-drivers

[51] Microsoft。《在 AKS 中使用 Azure Disk 创建和管理 PersistentVolume》。更新于 2026 年 6 月 3 日。https://learn.microsoft.com/en-us/azure/aks/create-volume-azure-disk

[52] Microsoft。《使用应用路由附加组件的托管 NGINX Ingress》。更新于 2026 年 8 月 3 日。https://learn.microsoft.com/en-us/azure/aks/app-routing

[53] Microsoft。《通过应用路由附加组件，使用 Kubernetes Gateway API 配置 Ingress》。更新于 2026 年 7 月 1 日。https://learn.microsoft.com/en-us/azure/aks/app-routing-gateway-api

[54] Microsoft。《AKS 网络资源最佳实践》。更新于 2026 年 4 月 13 日。https://learn.microsoft.com/en-us/azure/aks/operator-best-practices-network

[55] Microsoft。《AKS 部署与集群可靠性最佳实践》。https://learn.microsoft.com/en-us/azure/aks/best-practices-app-cluster-reliability

[56] Microsoft。《在 Azure Kubernetes 服务中创建节点池》。更新于 2026 年 7 月 13 日。https://learn.microsoft.com/en-us/azure/aks/create-node-pools

[57] Microsoft。《在 Azure Kubernetes 服务中使用系统节点池》。更新于 2026 年 4 月 30 日。https://learn.microsoft.com/en-us/azure/aks/use-system-pools

[58] Microsoft。《在 AKS 中使用适用于 Secrets Store CSI 驱动的 Azure Key Vault 提供程序》。更新于 2026 年 5 月 5 日。https://learn.microsoft.com/en-us/azure/aks/csi-secrets-store-driver

[59] Alibaba Cloud。《什么是容器服务 Kubernetes 版（ACK）？》更新于 2026 年 6 月 18 日。https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/product-overview/what-is-ack

[60] Alibaba Cloud。《ACK 集群概述》。更新于 2026 年 5 月 20 日。https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/user-guide/ack-cluster-overview/

[61] Alibaba Cloud。《节点池概述》。更新于 2026 年 7 月 9 日。https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/user-guide/node-pool-overview

[62] Alibaba Cloud。《ACK 用户指南》。更新于 2026 年 6 月 15 日。https://www.alibabacloud.com/help/en/ack/user-guide/

[63] Alibaba Cloud。《ACK 多租户安全》。更新于 2026 年 3 月 26 日。https://www.alibabacloud.com/help/en/ack/ack-managed-and-ack-dedicated/security-and-compliance/multi-tenancy-security

[64] Tencent Cloud。《TKE 概述》。https://www.tencentcloud.com/document/product/457/51208

[65] Tencent Cloud。《普通集群概述》。https://www.tencentcloud.com/document/product/457/6759

[66] Tencent Cloud。《节点池概述》。更新于 2026 年 3 月 13 日。https://www.tencentcloud.com/document/product/457/35900

[67] Tencent Cloud。《在 TKE 上部署 NGINX Ingress》。https://www.tencentcloud.com/document/product/457/38072

[68] Tencent Cloud。《TKE PersistentVolume 的 CBS 使用说明》。https://www.tencentcloud.com/document/product/457/36157

[69] Tencent Cloud。《TKE 计费概述》。https://www.tencentcloud.com/document/product/457/45157

[70] Volcengine。《什么是火山引擎容器服务（VKE）？》更新于 2026 年 4 月 15 日。https://docs.volcengine.com/docs/6460/70611?lang=en

[71] Volcengine。《VKE 集群概述》。更新于 2026 年 7 月 24 日。
https://docs.volcengine.com/docs/6460/1374026?lang=en

[72] Volcengine。《VKE 与其他云服务的依赖关系》。
https://docs.volcengine.com/docs/6460/100914?lang=en

[73] Volcengine。《使用 kubectl 创建 APIG Ingress》。
https://docs.volcengine.com/docs/6460/1327566?lang=en

[74] Pan, Wuming。《云与云 AI》第 7 讲：CaaS 实践。课程幻灯片，2025。

[75] Pan, Wuming。《云与云 AI》第 8 讲：工作负载与云持久化。课程幻灯片，2025。

[76] Pan, Wuming。《云与云 AI》第 9 讲：云应用定义。课程幻灯片，2025。

第三篇 / 第 12 章

持续交付、GitOps 与变更治理

不可变制品、版本化期望状态、拉取式协调、策略门禁、渐进式交付、遥测和回滚如何把变更转化为受治理的云资源

现代交付并不是更快的复制操作，而是一门控制平面学科：它连接意图、源、制品身份、环境策略、协调、运行时证据和恢复。GitOps 通过让期望状态具备声明式、版本化、自动拉取和持续协调的特征，强化了这门学科。由此形成的变更系统可以在执行前接受审查，在发布期间受到观测，并可依据证据撤销或修复。

本章内容

- 把软件交付视为受治理的反馈回路，而不是一系列构建与部署脚本。
- 根据每项实践所产生的状态及其自动化的决策，区分持续集成、持续交付、持续部署、发布和 GitOps。
- 只构建一次不可变制品，通过更改版本化期望状态，在各环境间提升同一摘要。
- 把环境建模为策略、身份、数据、SLO、配额、审批和证据边界，而不仅是分支名称或命名空间。
- 应用 OpenGitOps 原则：声明式期望状态、版本化且不可变的历史、自动拉取和持续协调。
- 比较 Flux 与 Argo CD 作为 Kubernetes 交付控制平面的特点，并区分托管 GitOps 服务与流水线驱动的集群部署。
- 使用拉取请求、CODEOWNERS、规则集、签名提交、必需检查、环境审批和例外记录作为变更治理控制。
- 分离开发者、构建器、签名者、发布者、GitOps 控制器和工作负载身份；在支持时使用工作负载联合，而不是长期云机密。
- 推理漂移、服务器端字段所有权、动态控制器、生成字段和协调冲突。
- 根据失败成本、可观测性和回滚约束，选择滚动、蓝绿、金丝雀、实验与渐进式交付策略。

- 区分运行时撤销、源代码还原和向前修复，尤其是在数据架构或不可逆副作用使简单回滚变得不安全时。
- 创建发布证据包，连接源、制品、配置、审批、已部署修订版、遥测、成本和结果。
- 把变更治理扩展到模型、提示词、检索索引、工具、安全策略、评估和智能体定义。
- 完成一项独立 AKS 与 Flux 实验，只通过更改 Git 中的期望状态执行一次发布和回滚。

云平台使基础设施和应用状态可编程，但可编程本身不会让变更变得安全。一次门户点击、流水线作业、Git 提交、Kubernetes 发布或智能体生成的变更都可能修改生产环境。因此，工程问题不只是如何自动化部署，而是如何让每项重要变更都可归因、可审查、可测试、符合策略、可观测、可逆转，并在经济上可问责。

传统部署过程经常把多项责任混在一个高权限脚本中：构建代码、选择版本、向集群进行身份验证、修改运行时对象、等待健康状态，以及判断结果是否可以接受。这种设计容易演示，却难以治理。脚本身份不断积累宽泛权限，运行时状态成为所发生事件的唯一记录，回滚则依赖仍然记得上条命令的人。

持续交付把这些责任分开。持续集成产生经过测试的不可变制品；提升决策改变版本化期望状态；协调器使环境趋向该状态；运行时遥测和业务证据决定变更应继续、停止、还原还是修复。对于声明式系统，GitOps 是这种模型特别强的一种形式，因为期望状态已版本化，而且控制器持续将其与实际状态比较。

核心主张。持续交付是一门控制平面学科。每项变更都应具有来源、不可变主体、审查与策略证据、特定环境授权、发布策略、遥测和恢复路径。GitOps 通过让期望状态具备声明式、版本化、自动拉取与持续协调的特征，强化了这份契约。

12.1 变更是一幅受管理的资源图

发布常用 3.7.2 之类的版本字符串表示。该标签便于人类理解，却不足以治理。生产状态由多个主体组成：源代码、构建指令、依赖、容器镜像或软件包、基础设施定义、应用配置、机密或机密引用、策略版本、数据库架构、模型部署、检索索引和运行时控制器状态。因此，发布是一幅图，而不是一个文件。

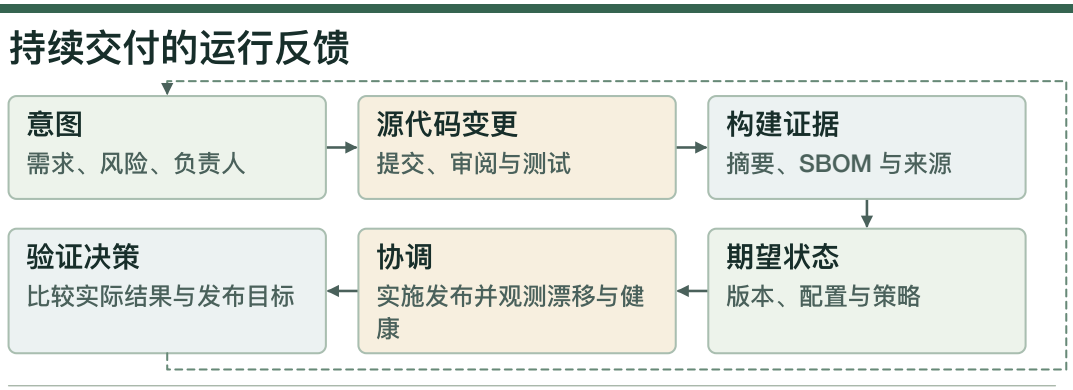


图 12-1 持续交付是从意图到已验证结果的受管理反馈回路。

表 12-1 受治理的变更图分离经常被合并为一次“部署”的主体。

主体	稳定标识符	控制问题	证据
变更意图	问题单、需求、风险记录或获准计划。	为何需要变更、由谁负责、预期取得什么结果？	所有者、作用域、风险等级、验收标准、到期时间。
源修订版	提交 SHA 或签名源归档摘要。	哪些代码和测试产生了候选版本？	存储库、作者、审查、签名、时间戳。
构建执行	工作流或流水线运行 ID。	由谁或什么对象、在哪个隔离环境中构建了制品？	构建器身份、输入、日志、测试结果、出处。
制品	软件包或镜像摘要。	正在提升哪些不可变字节？	摘要、SBOM、签名、漏洞与策略结果。
期望状态	配置提交、签名 OCI 配置摘要或发布清单。	环境应运行哪个制品、哪些设置与资源？	差异、策略结果、批准人、目标环境。
协调	控制器源修订版与应用状态。	运行时是否已趋向已声明状态？	GitRepository/Kustomization 或 Application 状态、事件、错误。
运行时状态	Deployment 修订版、Pod imageID、模型/索引版本、资源图。	实际正在运行并提供流量的是什么？	发布条件、对象清单、遥测、审计日志。
结果	评估窗口和决策 ID。	变更是否改善或维持了可靠性、安全、成本与业务价值？	SLI/SLO、错误预算、成本、安全发现、KPI、批准或回滚。

这幅图解释了为何“流水线成功”是薄弱证据。即使集群拒绝清单、新 Pod 无法通过就绪检查、制品摘要不同于获准主体、数据库迁移不兼容，或服务通过技术健康检查却违反业务 KPI，流水线仍然可能完成。受治理的发布记录必须越过自动化状态，延伸到实际运行时

与结果证据。

这幅图还阐明了控制平面边界。源代码存储库、制品注册表、部署控制器、策略引擎、环境审批、可观测性系统和成本管理服务，都是独立的资源系统。交付架构通过稳定标识符和明确的信任关系将它们组合起来。当任何一条边是隐式的——例如“latest 镜像”、共享管理员令牌或未记录的人工热修复——证据链就会断裂。

12.2 持续集成、持续交付、持续部署与发布

交付领域的术语经常被混用。精确用词很重要，因为每个术语都意味着不同的自动化边界和人工决策。持续集成主要处理从源到制品的路径。持续交付使软件保持可部署状态，并自动化产生发布候选版本的路径，同时在需要时保留明确的发布决策。持续部署进一步扩展自动化，使符合条件的变更无需例行人工门禁即可发布。发布则是让某个版本可供特定受众或工作负载使用的行为，可以与部署分开。

表 12-2 交付术语描述不同的状态转换。

术语	主要状态转换	典型自动化证据	人工决策
持续集成	源修订版 → 经过测试的构建候选版本。	编译、单元/集成测试、静态分析、依赖检查。	合并批准和失败检查的修复。
持续交付	获准候选版本 → 可发布制品与环境变更计划。	制品发布、部署计划、策略检查、分阶段发布就绪情况。	生产发布或环境审批仍可保持显式。
持续部署	符合条件的期望状态变更 → 自动进行生产发布。	全部持续交付证据，加上自动提升与验证。	人工定义策略和例外，而不是批准每次例行发布。
部署	把制品/配置置入运行时环境。	控制器状态、资源变更、发布条件。	可以自动、按计划或手动启动。
发布	已部署能力对用户或调用者开放。	流量规则、功能标志、API/模型部署激活、受众分配。	产品、风险、运维或自动化策略决策。
GitOps	版本化声明式期望状态 → 持续协调的实际状态。	源修订版、协调状态、漂移修正、控制器事件。	通过事实来源工作流与策略执行变更审批。

部署与发布可以解耦。新二进制文件可以部署在关闭的功能标志后，新模型部署可以不接收流量，蓝环境也可以在切换路由前预热。这种分离缩小了变更半径，却也产生更多必须治理的资源：功能标志、流量规则、受众策略、模型路由和休眠修订版的退役。

持续部署并非“无控制的持续变更”，而是以明确的自动化控制取代例行人工判断。如果证据薄弱、策略宽松或回滚不安全，增加自动化只会增加风险。成熟度问题不在于团队能多频繁地触发部署，而在于系统能否以该频率作出可信决策。

12.3 一次构建，提升同一制品

健全的交付系统只构建一次制品，以不可变方式标识它，并在各环境间提升同一主体。按环境重新构建会让同一源产生不同字节，削弱可追溯性；生产还可能依赖稍后的软件包解析、已变化的基础镜像，或一台未参与测试却遭入侵的构建运行器。

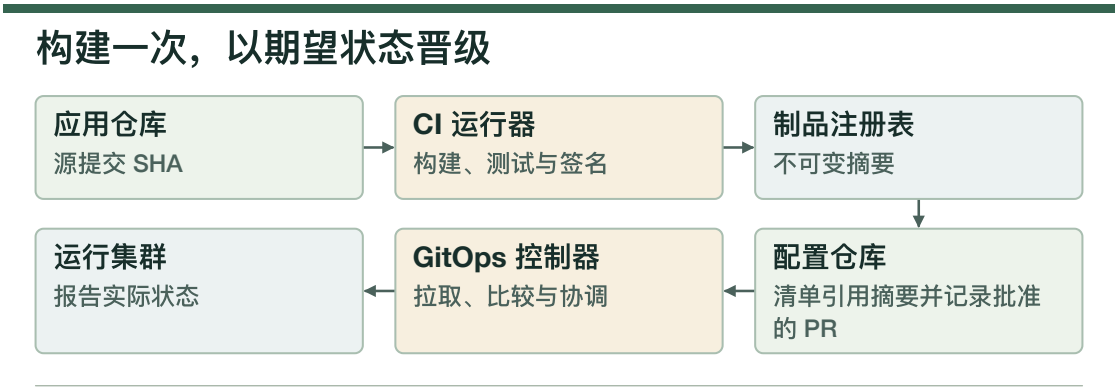


图 12-2 CI 产生不可变制品；GitOps 通过版本化期望状态提升制品。

提升应修改环境意图，而不是制品内容。配置变更可以更新镜像摘要、Helm Chart 版本、Kustomize 补丁、模型部署 ID 或基础设施模块版本。拉取请求应明确显示新旧主体。标签可保留为方便识别的名称，但可部署契约应解析为摘要或其他不可变标识符。

表 12-3 发布身份应当比人类可读版本更强。

字段	用途	示例
源修订版	把发布连接到源与审查历史。	Git 提交 SHA。
构建运行	标识构建器、工作流、时间与日志。	GitHub Actions 运行 ID 或 Azure Pipeline 运行 ID。
制品主体	为不可变软件包或镜像命名。	OCI 摘要 sha256:...，而不只是 :1.4.0。
SBOM/出处	描述组件与制品的生成方式。	SPDX/CycloneDX SBOM 与 in-toto/SLSA 出处。
配置修订版	为期望状态提交或 OCI 制品命名。	clusters/prod/kustomization.yaml 的 Git SHA。
环境	把决策绑定到目标策略与受众。	prod-eu、finance-staging 或租户组。
已部署身份	证明运行时解析到预期主体。	Kubernetes Pod imageID 摘要或模型部署版本。
决策窗口	定义评估遥测的时段。	2026-08-26T10:00Z 至 10:30Z。



图 12-3 发布是一幅最终落到实测结果的证据图。

相同的身份模型也适用于 AI 原生应用。模型名称不是部署身份；检索索引别名不是不可变索引构建；提示词标签不是提示词修订版；智能体版本也不是模型、知识、工具和安全依赖的完整集合。第十七章至第二十章分别定义这些资源。交付治理必须把它们绑定到发布台账中。

12.4 环境是策略边界

环境常被实现为资源组、云账户、订阅、Kubernetes 集群、命名空间或项目。任何此类容器都不能单独定义环境契约。生产环境是应用于某种目标状态的身份、网络、数据、配额、策略、审批、SLO、可观测性、保留和成本问责规则的组合。

表 12-4 环境设计应明确策略差异。

维度	开发	预生产	生产
目的	快速反馈与本地集成。	在类似生产的约束下验证发布候选版本。	服务已承诺用户与业务结果。
身份	开发者在有界作用域内自助服务。	流水线/协调器身份，并允许测试运维人员访问。	专用协调与工作负载身份；紧急访问保持分离。
数据	合成或最小化数据。	有代表性、已脱敏或受控的数据。	具有保留、隐私、备份和恢复规则的权威数据。
网络	可以允许更宽泛的调试访问。	类似生产的私有路径与入口策略。	获准暴露、出口控制、私有依赖、受监控路径。

维度	开发	预生产	生产
变更门禁	自动化测试和所有者审查。	发布审查、策略检查、评估套件。	根据风险等级要求批准或使用自动化策略。
SLO/证据	工程诊断。	发布资格认定以及负载/评估证据。	面向用户的 SLO、错误预算、安全、成本、业务 KPI。
经济性	低成本、可弹性调整或临时资源。	用于测试的代表性规模；安排明确生命周期。	容量计划、承诺、异常控制、成本所有权。

分支不是环境。分支可以表示一条意图流，却不会创建隔离凭据、网络策略、数据控制、配额或 SLO。同样，命名空间也不会自动成为环境：集群级控制器、共享节点、共享网关、共享机密和提供商资源都可能跨越命名空间边界。环境必须由可强制执行的控制和证据定义。

提升应保留制品身份，同时应用特定环境的配置契约。配置可以因环境而有合理差异，但差异必须明确且可审查。机密值不应存储在 Git 中；存储库应包含引用、身份和策略，使系统在运行时获得正确机密。环境覆盖层应足够精简，让审查者能够理解有效状态，而不必在头脑中执行迷宫般的补丁。

12.5 GitOps 原则与协调契约

OpenGitOps 定义四项原则：期望状态是声明式的；历史经过版本化且不可变；软件代理自动拉取期望状态；代理持续把实际状态向期望状态协调。这些原则比任何特定存储库布局或控制器产品都更重要。团队可能使用时髦工具却错误实施 GitOps，也可以通过朴素控制器和有纪律的事实来源工作流良好应用这些原则。

表 12-5 OpenGitOps 原则带来具体的工程义务。

原则	工程义务	缺失时的失败
声明式	声明应当存在什么，而不是一次性命令序列。	系统无法可靠检测或修复漂移。
版本化且不可变	保留完整历史与稳定修订版身份。	审计人员和运维人员无法复现或解释过去状态。
自动拉取	由代理从源获取获准的期望状态。	外部部署器需要宽泛运行时凭据，并成为可用性依赖。
持续协调	反复比较期望状态与实际状态并采取行动。	人工变更与局部失败持续存在而无人察觉。

Git 是最常见的源，因为它提供审查、历史、签名、分支和熟悉的协作方式。更深层的契约是版本化期望状态。Flux 还可以从 OCI 制品及其他源协调期望状态。基于 OCI 的配置源可以降低运行时对 Git 服务器的依赖，并把清单、签名、SBOM 和镜像置于共同的内容寻址分发系统中。即使生产源是签名 OCI 制品，团队仍应保留由人员治理的创作历史。

GitOps 并不意味着每个字节都属于 Git。机密、大型模型文件、训练数据、数据库内容、运行时指标和生成状态，应存放在为其设计的系统中。Git 存储声明、引用、策略与版本化元数据，使控制器能够安全获取或生成这些主体。该区别与本书一贯采用的模型相同：期望状态属于控制平面；运维载荷属于合适的数据平面。



图 12-4 拉取式 GitOps 把部署权限移交给集群侧协调器。

只有当权限得到收窄时，拉取模型才是安全改进。一个能够读取所有存储库、解密所有机密并修改所有命名空间的集群级协调器，仍是高价值主体。存储库凭据、控制器 ServiceAccount、源允许列表、命名空间边界、网络出口、准入策略和多租户控制都需要显式设计。拉取式交付移除了一条危险的凭据路径，却不会消除身份工程。

12.6 Flux 与 Argo CD 作为交付控制平面

Flux 和 Argo CD 都是 CNCF 毕业项目，也都为 Kubernetes 实现声明式、拉取式协调，但二者的对象模型与运维体验有所不同。正确的比较不是哪个 UI 更好看，而是各自如何表示源、应用、依赖、健康、策略、多租户和证据。

表 12-6 Flux 与 Argo CD 公开不同但可比较的交付抽象。

关注点	Flux	Argo CD
源	GitRepository、OCIRepository、Bucket、HelmRepository 以及生成的制品。	由 Application/ApplicationSet 引用的存储库或 Chart 源。
应用单位	具有依赖与健康检查的 Kustomization 和 HelmRelease 资源。	具有源、目标、同步策略、健康与历史的 Application 资源。

关注点	Flux	Argo CD
协调风格	由源、Kustomize、Helm、通知和镜像自动化控制器组成。	以 Application 为中心的控制器和 API/UI，支持自动或手动同步。
多集群	根据设计，由一个或多个控制器/配置以本地或远程集群为目标。	ApplicationSet 与集群注册支持集群舰队模式。
漂移/自愈	Kustomize 和 Helm 控制器依据配置检测并修正漂移。	自动同步可以执行修剪与自愈；同步窗口和资源自定义治理其行为。
渐进式交付	与 Flagger 或其他发布控制器集成。	与 Argo Rollouts 自然集成，执行金丝雀/蓝绿分析。
运维界面	Kubernetes CRD、Flux CLI、事件、指标与通知。	Kubernetes CRD，加上 CLI、API 与功能丰富的 Web UI。

Azure 把 Flux 作为托管集群扩展公开，并将每项配置表示为 Azure 资源。microsoft.flux 扩展安装源、Kustomize、Helm 和通知控制器。Microsoft.KubernetesConfiguration/fluxConfigurations 资源描述源与协调参数，状态则回传到 Azure。这种组合很重要：Kubernetes CRD 执行协调，Azure Resource Manager 为扩展和配置资源提供生命周期、身份、大规模策略、清单与活动证据。

阿里云 ACK One 舰队托管具备多集群能力的 Argo CD。它自动执行安装、升级和扩缩容，并集成阿里云 SSO 与用户权限。这是一层托管 GitOps 控制平面，而不仅是在流水线中运行 kubectl 的一个步骤。腾讯云 CODING 持续部署和火山引擎 CodePipeline 为向 TKE/VKE 与 Kubernetes 的流水线驱动部署提供强大支持，包括分阶段策略和审批。TKE 或 VKE 上严格的拉取式 GitOps 设计可以通过自主管理 Flux 或 Argo CD 实现。每种方式都可能让同一 YAML 到达集群，但凭据路径、协调行为、漂移修正和证据模型不同。

表 12-7 托管 GitOps 与流水线驱动部署在运维上并不等价。

维度	托管拉取式 GitOps	流水线驱动的推送部署
运行时权限	集群侧控制器持有受限的应用权限。	外部流水线身份向集群或部署终结点进行身份验证。
事实来源	版本化期望状态得到持续协调。	流水线运行应用一个快照；除非另行检测，后续漂移可能持续存在。
集群暴露	控制器主动访问外部源；构建运行器不需要集群终结点。	流水线必须能到达集群 API 或提供商部署服务。
故障恢复	控制器重试，并在暂时性故障后趋于一致。	流水线可能需要重新运行；局部故障后的状态取决于工具。
漂移	一种一等的观测差异，可以得到修正。	通常由独立工具或后续部署检测。

维度	托管拉取式 GitOps	流水线驱动的推送部署
证据	源修订版、协调条件与控制器事件。	流水线日志、部署任务结果与集群状态。

12.7 Git 中的变更治理

Git 提供历史记录，但本身并不提供治理。允许直接强制推送到生产分支的存储库虽然版本化，却控制薄弱。变更治理围绕事实来源路径增加身份、审查、验证、决策权限和例外处理。



图 12-5 变更治理是一系列产生证据的门禁。

表 12-8 存储库控制映射到不同的治理目标。

控制	目标	证据
拉取请求要求	防止未经审查直接更改受保护的期望状态。	PR 编号、作者、差异、关联问题单。
必要批准	应用职责分离和基于风险的审查。	审查者身份与批准时间戳。
CODEOWNERS 或路径审查者	把敏感目录路由给负有责任的团队。	平台、安全、数据或模型所有者的批准。
必需状态检查	在测试、策略、清单渲染、签名或评估通过前阻止合并。	与提交 SHA 关联的具名检查结果。
取消过期批准	确保获准差异就是最终合并的差异。	变更后使审查失效，并重新批准。
签名提交/标签	在信任模型支持时，加强作者与发布身份。	验证状态与签名密钥/证书。
线性历史或受约束合并	简化出处和自动化策略评估。	受保护分支规则与由此形成的提交图。

控制	目标	证据
环境批准	在合并或提升请求后，授权特定于生产环境的决策。	批准人、环境、制品/配置修订版、到期时间。
例外记录	允许具有时限、明确所有者且具备补偿控制的偏离。	理由、风险所有者、作用域、到期时间与修复计划。

审查质量取决于差异。把清单中的镜像标签从 latest 改为 latest，什么也没有揭示。把一个不可变摘要改为另一个摘要，并随附源修订版、SBOM、漏洞差异、发布计划和评估结果，才让审查者获得具体主体。生成的清单应在 CI 中渲染，使审查者能看到有效状态；生成器输入则仍是权威来源。

批准数量应随风险而定，而不是随仪式而定。只读可观测性仪表板变更不应与生产数据库迁移共用完全相同的门禁。风险分类可以考虑影响范围、数据敏感度、可逆性、用户影响、新颖性和自动化证据强度。高风险变更可能需要平台、安全、数据或模型所有者审查，并要求预演已定义的回滚或向前修复方案。

智能体与代码生成系统让归因更加重要。即使由 AI 系统创建补丁，负有责任的人员或服务所有者仍应清晰可见。存储库规则可以要求未经归因的自动拉取请求接受额外审查。治理原则很简单：自动化可以准备变更，但合并或发布权限必须明确分配且可审计。

12.8 身份、机密与职责分离

交付系统连接多个高价值信任域。所有步骤共用一个服务主体或 kubeconfig，架构图很简单，却难以防护。应根据每个身份执行的操作分离身份。

表 12-9 交付身份链应让责任保持狭窄。

主体	所需权限	不应拥有的权限
开发者	创建分支和拉取请求；读取非敏感发布证据。	直接变更生产集群或读取生产机密。
CI 构建器	读取源、获取依赖、运行测试、发布候选制品与证据。	生产部署权限或宽泛存储库管理权限。
制品签名者/证明者	在受保护工作流条件下为已验证主体签名。	任意修改源或访问运行时管理员。
配置更新者	提出或合并获准制品身份到环境路径的变更。	重新构建或替换制品字节。
GitOps 协调器	读取获准期望状态，并把有界资源应用到目标作用域。	存储库写入、无关命名空间或不受限的云订阅访问。
发布批准者	批准特定环境、主体和时间窗口。	批准后悄然修改候选版本。

主体	所需权限	不应拥有的权限
工作负载身份	访问应用所需的运行时数据平面依赖。	存储库、CI 或 GitOps 管理权限。
紧急操作员	具有时限、增强日志记录的紧急干预。	永久例行使用或未经审查的凭据共享。

GitHub Actions 和其他现代 CI 系统可以使用 OpenID Connect 联合，把工作流身份交换为短期云令牌，从而避免在存储库中存放长期云客户端机密。云信任策略应约束存储库、分支或环境、工作流、受众与主体声明，由此产生的云角色仍应保持最小权限。OIDC 移除了已存储机密，却不能成为宽泛授权的理由。

拉取式 GitOps 进一步分离身份。CI 运行器发布制品并提出期望状态，集群内控制器应用状态。这样可以防止 CI 系统成为通用集群管理员，并允许集群访问保持私有。但协调器可能需要源凭据、解密能力、注册表访问或云身份。这些凭据需要轮换、出口控制、审计与命名空间隔离。

机密应通过引用和工作负载身份进入运行时，而不是出现在渲染后的明文清单或 CI 日志中。使用机密同步控制器时，其源、目标命名空间、刷新间隔、删除语义和控制器权限都成为交付契约的一部分。同步成功并不能证明工作负载具备正确的最小权限数据平面权限。

12.9 漂移、字段所有权与动态控制器

持续协调让漂移可见且可修正，但并非每项运行时差异都是未经授权的漂移。Kubernetes 控制器会合理地修改状态、分配地址、注入默认值、更新副本数、轮换证书并附加提供商资源。GitOps 系统必须区分已声明字段、由其他控制器拥有的字段以及平台生成的字段。

表 12-10 不同漂移类别需要不同响应。

漂移类型	示例	正确响应
未经授权的命令式漂移	运维人员直接编辑镜像、标签或安全上下文。	协调回源状态、发出警报并调查绕过路径。
控制器拥有的动态字段	HPA 更改副本，或 Service 控制器分配地址。	排除或忽略该字段；不要让控制器相互争夺。
默认化或变更的字段	准入过程添加默认值、Sidecar、标签或安全设置。	渲染/理解有效状态；如果差异稳定，则配置差异自定义。
生成的机密或证书	控制器创建轮换凭据。	存储生成器意图，而不是临时值；治理生成器。

漂移类型	示例	正确响应
紧急运行时变更	事件期间的紧急缓解措施。	记录并设定时限，随后提交权威源，或允许协调将其移除。
外部依赖漂移	DNS、数据库、模型别名或云策略在存储库外发生变更。	关联提供商证据，并决定是否需要扩展期望状态模型。

字段所有权应明确。如果 HPA 拥有副本，除非架构有意禁用 HPA，否则 Git 存储库不应反复强制固定副本数。如果策略控制器注入标签，GitOps 差异应理解这种变更。如果数据库 Operator 管理子资源，应用团队应声明 Operator 自定义资源，而不是声明每个生成的 StatefulSet。事实来源不一定是一个文件，而是权威声明与控制器契约的集合。

暂停也是一种受治理状态。Flux 与 Argo CD 可以暂停协调或自动同步。在事件响应、迁移或调试期间可能需要暂停，但它会禁用限制漂移的机制。该操作应有时限、获授权、可观测，并在之后接受协调审查。永久暂停的 GitOps 对象其实是带有误导标签的非托管环境。

12.10 发布策略与渐进式交付

部署控制器可以逐步替换副本，但滚动更新不会自动成为安全发布策略。正确的策略取决于容量、兼容性、故障检测、流量控制、数据状态，以及获得可信证据所需的时间。

表 12-11 发布策略以不同方式权衡容量、控制与证据。

策略	机制	优势	主要风险
重建	先停止旧修订版，再启动新修订版。	简单，适合排他或不兼容的工作负载。	有意停机，无法实时比较。
滚动更新	在激增/不可用限制内逐步替换旧副本。	原生、经济，适合兼容的无状态服务。	版本混合，流量/证据控制有限。
蓝绿	准备完整的并行环境，再切换流量。	快速切换，并可迅速反向切回流量。	双倍容量、状态兼容性、陈旧后台工作。
金丝雀	先暴露给少量受众或流量比例，再逐步增加。	限制影响范围，并支持基于证据的决策。	需要可靠的流量控制、分群和分析。
实验/A-B	把指定群组路由到不同方案以比较结果。	度量业务或模型行为，而不仅是技术健康。	统计偏差、用户一致性和数据治理复杂性。
影子	把流量复制给候选版本，但不提供其响应。	在较低用户影响下验证负载与行为。	除非请求已脱敏，否则可能造成重复副作用以及隐私/成本风险。

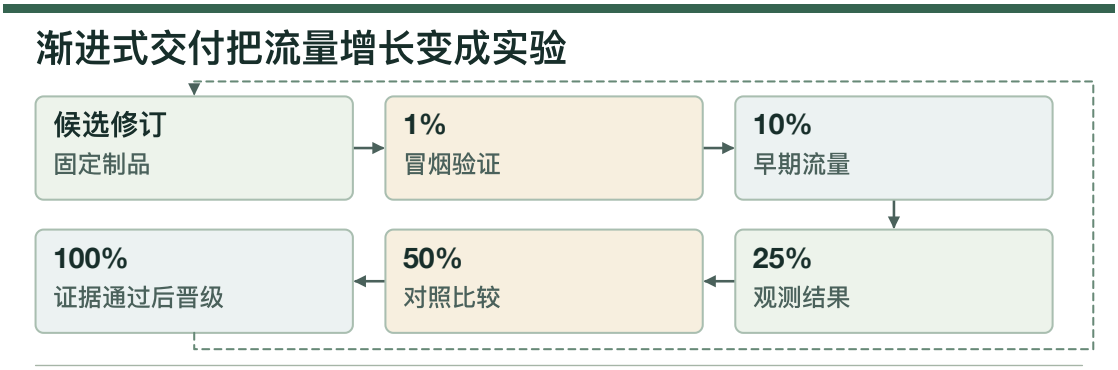


图 12-6 只有证据持续符合策略，渐进式交付才推进流量。

Argo Rollouts 为蓝绿和金丝雀策略、加权流量切换、人工判断、指标分析、自动提升与回滚提供 Kubernetes 自定义资源。Flagger 或云原生部署服务也可实现类似行为。产品不如契约重要：每个提升步骤都应标识候选版本、流量群组、度量窗口、阈值、决策权限和回退方案。

指标必须反映用户体验。CPU 利用率可以表示饱和度，却很少能证明发布安全。渐进式策略应结合可用性、延迟、错误率、依赖健康、安全信号、成本，以及相关的业务或模型质量。对于 AI 应用，分析可以包括依据性、检索精确率、安全策略违规、工具调用成功率、每项结果的令牌成本和人工升级率。

发布控制器不应以自动回滚掩盖不确定性。稀疏流量、延迟故障、缓存、异步作业或季节性行为都可能让度量窗口无法得出结论。“没有超过阈值”并不等于“候选版本更好”。策略应允许无定论结果暂停提升并请求人工判断。

12.11 回滚、向前修复与状态兼容性

回滚常被描述为持续交付的通用安全机制。只有当上一版本仍与当前状态及外部影响兼容时，回滚才安全。无状态代码通常可以快速还原；数据库架构变更、消息格式、数据迁移、模型输出持久化、索引重写和不可逆外部操作，则可能需要向前修复或分阶段兼容性计划。



图 12-7 运行时撤销、源还原与向前修复是不同的恢复操作。

表 12-12 恢复选择取决于权限与兼容性。

情形	首选操作	原因
清单或镜像回归；旧版本仍兼容。	还原期望状态提交，让控制器协调。	同时恢复权威历史与运行时。
存储库工作流完成前发生严重紧急事件。	执行有界运行时撤销或流量切换，随后立即更新源。	迅速减轻损害，同时避免永久非托管漂移。
已经迁移了不向后兼容的数据库架构。	执行向前修复或扩展/收缩迁移计划。	旧代码可能在新架构上损坏数据或失败。
配置错误，但制品无缺陷。	只还原配置提交。	保留有效制品身份，并缩小变更范围。
候选版本在完全提升前违反阈值。	中止候选版本，并把流量恢复到稳定修订版。	限制影响范围；无需回滚整个环境。
安全入侵或不可信制品。	阻止该主体、撤销凭据、隔离、部署可信替代版本，并调查证据。	常规回滚可能重新引入另一个易受攻击的主体，或丢失取证状态。

Kubernetes 为受支持的工作负载控制器提供 kubectl rollout undo。在 GitOps 管理的环境中，该命令改变运行时状态，却不会更改权威源。持续协调的控制器可能重新应用当前 Git 修订版。因此，运行时撤销是紧急缓解措施，而不是最终的 GitOps 回滚。持久操作是还原或创建新提交，表达预期恢复状态。

Argo CD 还增加了一项细节：自动同步会与回滚相互影响。当前 Argo CD 文档指出，当启用自动同步时，无法对 Application 执行回滚。团队应提前定义运维过程——还原 Git、在受控权限下禁用自动化，或使用渐进式交付流量控制——而不是在事件期间才发现该约束。

迁移应内置向前兼容性。扩展/收缩数据库变更、宽容读取器、版本化消息、双写转换、模型路由别名和索引别名，可以让新旧组件共存。这些技术把回滚从一厢情愿变成可执行契约。

12.12 遥测、证据与交付绩效

交付系统既需要发布证据，也需要长期绩效证据。发布证据回答变更了什么、谁授权、部署了什么，以及即时结果是否符合策略。绩效证据回答交付系统是否随着时间改善速度、稳定性、恢复与返工。

表 12-13 最小发布证据包把意图连接到结果。

证据组	所需记录
意图与源	问题单/变更 ID、所有者、风险等级、源提交、拉取请求、审查者。
构建与制品	工作流运行、测试结果、制品摘要、SBOM、出处、签名、扫描差异。
期望状态	配置提交/摘要、渲染后差异、环境、策略检查、批准。
协调	控制器源修订版、健康条件、应用/修剪结果、控制器事件。
运行时	Deployment/发布修订版、Pod imageID 或服务/模型版本、资源图、流量划分。
结果	SLI/SLO 窗口、错误预算影响、安全事件、成本差异、业务或 AI 质量 KPI。
恢复	回滚/向前修复计划、执行记录、最终权威修订版、事件关联。
保留与完整性	存储位置、访问策略、哈希/签名、保留期、删除权限。

当前 DORA 软件交付绩效模型使用五项指标。吞吐量由变更前置时间、部署频率和失败部署恢复时间表示；不稳定性由变更失败率和部署返工率表示。这些指标应应用于应用或服务层级，结合上下文解释，并用于改进，而不是让不同团队之间进行竞争。

表 12-14 交付指标需要结合上下文解释。

指标	问题	常见误用
变更前置时间	一项已接受变更需要多长时间才能达到已定义生产状态？	不同团队采用不同的计时起止点。
部署频率	该服务多久向生产环境部署一次？	脱离结果或质量，把更多部署次数本身视为价值。
失败部署恢复时间	部署失败后，服务恢复需要多长时间？	度量事件关闭时间，而不是用户结果恢复时间。
变更失败率	有多大比例的部署因服务降级而需要修复？	以不一致的分类隐藏失败。
部署返工率	有多少部署活动用于修正先前的部署工作？	把有价值的迭代改进计为失败，或忽略不可见的人工返工。

交付遥测应与发布证据图关联。追踪和日志属性可以包括发布 ID、源修订版、制品摘要、配置修订版、环境、发布步骤，以及功能或模型版本。高基数标识符适合放在追踪或日志中，派生指标则支持趋势与 SLO 决策。第十三章定义使用这些信号的受管理反馈回路；本章定义发布身份如何进入该回路。

成本是发布证据的一部分。新版本可能可靠，却在经济上不可接受，因为它提高了每次请求的 CPU、模型令牌用量、检索调用、日志量或跨区域传输。第十四章的 FinOps 控制平面应比较变更前后每项成功结果的标准化成本。因此，回滚阈值既可以是技术性的，也可以是经济性的。

12.13 AI 原生变更治理

AI 原生系统扩展了发布主体。应用发布可能更改代码但不更改模型，更改模型但不更改代码，重建索引，更新提示词指令，授予新工具，调整安全阈值或修改智能体工作流。每项变更都可能独立影响行为与成本。单一应用版本已不足以解释生产行为。

表 12-15 AI 原生主体需要专门的门禁与证据。

主体	版本/证据	发布门禁
模型部署	提供商、模型/版本、部署 ID、容量模式、安全策略、评估结果。	质量、安全、延迟、配额、退役与成本阈值。
提示词/指令	不可变提示词修订版、模板变量、系统策略版本。	回归集、注入测试、风格/行为批准。
知识/索引	源快照、分块、嵌入模型、架构、索引构建、别名目标。	检索精确率/召回率、ACL 筛选、引用完整性、新鲜度与成本。
工具契约	工具架构、实现版本、身份、作用域、风险等级。	最小权限、验证、试运行、批准、幂等性与审计。
安全策略	策略版本、阈值、例外、评估器版本。	对抗测试、假阳性/假阴性审查、所有者批准。
智能体定义	模型、提示词、记忆、知识、工具、工作流、评估套件。	端到端任务成功率、追踪审查、过度自主性检查与结果成本。
评估套件	数据集/快照、评分规则、评判模型/版本、阈值。	数据集治理、可复现性、泄露与统计充分性。

提升应使用复合发布清单为每个主体命名。这并不要求一个单体文件，却要求一个可查询的发布身份。当智能体给出错误答案时，调查人员必须能够确定该次运行使用了哪个模型部署、知识别名、提示词修订版、工具契约、安全策略和代码制品。追踪与评估系统应记录这些标识符。

AI 变更可能需要更缓慢或分群的发布。新模型可以按租户、任务类型、地理区域或请求百分比进行金丝雀发布；新检索索引可以在别名切换前执行影子查询并比较；新工具可以从只读或试运行模式开始；智能体策略可以要求人工批准，直到有足够证据支持更高自主程度。因此，渐进式交付既是流量控制机制，也是自主性控制机制。

12.14 多云工程意图映射



图 12-8 不等价的多云交付控制平面可以产生相似部署结果。

表 12-16 参考云平台的交付能力。

工程意图	Azure	阿里云	腾讯云	火山引擎
代码与 CI	GitHub Actions 或 Azure Pipelines；OIDC/联合 Azure 登录。	阿里云云效/Codeup 与流水线。	CODING 存储库和 CI。	代码存储库与 CodePipeline。
制品注册表	具备 OCI 证据的 Azure Container Registry。	容器镜像服务企业版/个人版。	腾讯云容器镜像服务。	容器镜像仓库。
托管 Kubernetes	AKS。	ACK。	TKE。	VKE。
托管 GitOps	用于 AKS/Arc 的 microsoft.flux 扩展与 Azure fluxConfigurations。	ACK One 舰队托管具备多集群能力的 Argo CD。	文档介绍了流水线驱动的 CODING 持续部署；严格拉取式 GitOps 可在 TKE 上自主管理。	文档介绍了流水线驱动的 CodePipeline/VKE 部署；严格拉取式 GitOps 可在 VKE 上自主管理。
渐进式交付	Argo Rollouts/Flagger 或 Azure 原生流量与监控集成。	ACK One GitOps 配合 Argo Rollouts 执行金丝雀与分析。	CODING 部署策略，或 TKE 上自主管理的发布控制器。	批量/金丝雀流水线任务，或 VKE 上自主管理的发布控制器。

工程意图	Azure	阿里云	腾讯云	火山引擎
变更治理	GitHub 规则/环境、Azure DevOps 审批、Azure Policy、工作负载联合。	RAM/SSO、存储库/流水线审批、ACK/Cloud Config 治理。	CODING 审批与凭据、CAM、TKE 策略与可观测性。	流水线审批、IAM、VKE 策略与可观测性。
证据	Git/Actions 或 Pipelines、ACR、Azure 中的 Flux 状态、Kubernetes 状态、Azure Monitor。	存储库/流水线、注册表、托管 Argo CD 状态、ACK 与 ARMS/SLS。	CODING 交付记录、TCR、TKE 状态、CLS/监控。	流水线记录、注册表、VKE 状态、日志/监控。

本表是工程意图地图，并不声称产品等价。Azure 与 ACK One 提供托管拉取式 GitOps 控制平面；参考文档中的腾讯云 CODING 与火山引擎 CodePipeline 强调流水线驱动交付。四者都能运行 Kubernetes 工作负载，也都能托管 Flux 或 Argo CD，但默认信任路径、管理责任、多集群模型与证据界面有所不同。

可移植的配置格式不会创造可移植的运维。同一个 Kustomization 可以由托管 Flux、托管 Argo CD、自主管理控制器协调，或由流水线应用。团队应记录凭据位于何处、由谁负责控制器升级、如何处理漂移、如何强制执行多租户、哪些日志能证明协调，以及当源或提供商控制平面不可用时平台如何表现。

12.15 实验 12——仅通过更改 Git 中的期望状态执行发布与回滚

这项独立实验在 Azure Kubernetes 服务上证明 GitOps 核心契约。你将把一个小型 Kubernetes 应用置于版本化期望状态下，创建 Azure 托管 Flux 配置，通过在 Git 中更改镜像引用发布新应用版本，引入一次受控漂移，并通过 Git revert 回滚。权威发布或回滚都不使用 kubectl apply、kubectl set image 或直接集群变更。

实验验收标准。只有当证据包连接 Git 提交、Flux 源修订版、Kubernetes 发布状态、运行时镜像摘要、应用版本响应、漂移修正、回滚提交和清理时，实验才算完成。只有一条绿色命令并不足够。

12.15.1 能力目标与边界

- 主要能力。版本化期望状态、拉取式协调、漂移修正，以及以源为权威的回滚。
- **Azure** 资源。一个现有或临时的、使用托管身份的 AKS 集群，以及 Microsoft Flux 扩展/配置。
- 应用。Podinfo：一个小型 Kubernetes 参考服务，初始部署 6.14.0，随后更新到 6.14.1。

- **存储库**。实验使用一个公开 Git 存储库。私有存储库也可使用，但需要额外的源身份配置。
- **暴露**。ClusterIP 加 kubectl port-forward；实验不会创建公网负载均衡器。
- **成本边界**。AKS 集群与节点是主要成本。复用获准集群，或在实验后立即删除临时集群。
- **范围外**。构建应用镜像、签名制品、生产机密交付、多租户集群加固和高级流量划分。

12.15.2 先决条件

- Azure CLI，具备可用的 k8s-configuration 和 k8s-extension 命令；kubectl、Git，以及可以管理 AKS 集群与扩展的 Azure 订阅。
- 一个使用托管身份的 AKS 集群。如果没有保留其他练习的集群，请在专用资源组中创建一个小型临时集群，并记录其成本时段。
- 一个可推送的公开 Git 存储库。建议使用拉取请求规则保护 main 分支，但机械操作练习并不强制要求。
- 用于资源组、集群、存储库 URL、分支和 Flux 配置名称的本地变量。

```
export RG="book-gitops-rg"
export AKS="book-gitops-aks"
export REPO_URL="https://github.com/YOUR-ACCOUNT/book-gitops.git"
export BRANCH="main"
export FLUX_CONFIG="book-gitops"
export APP_NS="book-gitops"
```

CLI 语法与扩展版本会持续演进。执行前检查 az k8s-configuration flux create -h 以及当前 Azure GitOps 文档。这里使用的配置以 AKS managedClusters 资源为目标。

12.15.3 创建期望状态存储库

创建以下存储库结构。base 包含命名空间、Deployment 和 ClusterIP Service；cluster 路径组合 base。在更大的系统中，应更明确地分离环境覆盖层、策略对象与依赖。

```
book-gitops/
├── apps/
│   └── podinfo/
│       └── base/
│           ├── namespace.yaml
│           ├── deployment.yaml
│           ├── service.yaml
│           └── kustomization.yaml
└── clusters/
    └── book/
        └── kustomization.yaml
```

使用以下最小资源。练习中有意保留可见的镜像标签；在生产环境中，应解析并记录不可变摘要，将其作为提升证据的一部分。

```
# apps/podinfo/base/namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: book-gitops
  labels:
    app.kubernetes.io/part-of: book-gitops-lab
---
# apps/podinfo/base/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: podinfo
  namespace: book-gitops
spec:
  replicas: 2
  selector:
    matchLabels:
      app: podinfo
  template:
    metadata:
      labels:
        app: podinfo
    spec:
      securityContext:
        runAsNonRoot: true
      containers:
      - name: podinfo
        image: ghcr.io/stefanprodan/podinfo:6.14.0
        ports:
        - name: http
          containerPort: 9898
        readinessProbe:
          httpGet:
            path: /readyz
            port: http
        livenessProbe:
          httpGet:
            path: /healthz
            port: http
        resources:
          requests:
            cpu: 50m
            memory: 64Mi
          limits:
```

```
            cpu: 250m
            memory: 128Mi
---
# apps/podinfo/base/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: podinfo
  namespace: book-gitops
spec:
  selector:
    app: podinfo
  ports:
  - name: http
    port: 9898
    targetPort: http
---
# apps/podinfo/base/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- namespace.yaml
- deployment.yaml
- service.yaml
---
# clusters/book/kustomization.yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
resources:
- ../../apps/podinfo/base
```

提交并推送初始状态。把提交 SHA 记录为初始期望状态修订版。

```
git add .
git commit -m "Deploy podinfo 6.14.0"
git push origin main
git rev-parse HEAD | tee evidence/initial-config-sha.txt
```

12.15.4 创建 Azure Flux 配置

获取集群凭据以便观测。权威部署仍由 Flux 执行。

```
az aks get-credentials --resource-group "$RG" --name "$AKS" --overwrite-existing
```

创建 Flux 配置。创建第一个配置时，如果 Microsoft Flux 扩展尚不存在，将自动安装。Kustomization 启用修剪，因此从期望状态中移除的对象也会从集群中移除。

```
az k8s-configuration flux create   --resource-group "$RG"   --cluster-name
"$AKS"   --cluster-type managedClusters   --name "$FLUX_CONFIG"   --scope
cluster   --namespace flux-system   --kind git   --url "$REPO_URL"   --branch
"$BRANCH"   --sync-interval 1m   --kustomization name=book-app
path=./clusters/book prune=true timeout=2m
```

如果较新版本的 CLI 要求 Kustomization 参数以逗号分隔或重复提供，请使用帮助输出保持相同意图：名称为 book-app、路径为 ./clusters/book、prune 为 true、超时为两分钟。

```
az k8s-configuration flux show   --resource-group "$RG"   --cluster-name "$AKS"
--cluster-type managedClusters   --name "$FLUX_CONFIG"   --output json >
evidence/flux-configuration.json

kubectl -n flux-system get gitrepositories,kustomizations -o wide
kubectl -n "$APP_NS" get deployment,pods,service
kubectl -n "$APP_NS" rollout status deployment/podinfo --timeout=3m
```

记录源修订版和实际镜像摘要。imageID 字段可以证明运行时从人类可读标签解析出了哪个不可变镜像。

```
kubectl -n flux-system get gitrepository -o json   > evidence/git-source-
status.json
kubectl -n flux-system get kustomization -o json   > evidence/kustomization-
status.json
kubectl -n "$APP_NS" get pods -l app=podinfo   -o jsonpath='{range .items[*]}
{.metadata.name}{"  "}{.spec.containers[0].image}{" "}
{.status.containerStatuses[0].imageID}{"
"}{end}'   | tee evidence/runtime-initial.txt
```

转发一个本地端口并验证应用版本。如果愿意，可在第二个终端中运行端口转发。

```
kubectl -n "$APP_NS" port-forward service/podinfo 9898:9898 &
PF_PID=$!
sleep 3
curl -fsS http://127.0.0.1:9898/version   | tee evidence/version-initial.json
kill "$PF_PID"
```

12.15.5 通过 Git 发布新版本

只把镜像引用从 6.14.0 改为 6.14.1。在受治理的存储库中，应创建分支与拉取请求，运行必需检查，审查渲染后的差异并合并。对于个人实验存储库，直接提交在机械上足够，但治理证据较弱。

```
git checkout -b release/podinfo-6.14.1
sed -i.bak 's/podinfo:6.14.0/podinfo:6.14.1/'
apps/podinfo/base/deployment.yaml
rm apps/podinfo/base/deployment.yaml.bak
git diff
git add apps/podinfo/base/deployment.yaml
git commit -m "Promote podinfo to 6.14.1"
git push -u origin release/podinfo-6.14.1
# Merge the reviewed pull request, or merge locally for the lab.
git checkout main
git pull --ff-only
git rev-parse HEAD | tee evidence/release-config-sha.txt
```

等待 Flux 观测到该提交，并等待 Kubernetes 完成发布。为了加快实验，可以触发协调，但应用期望状态的参与者仍然是控制器。

```
flux reconcile source git book-gitops -n flux-system --with-source || true
flux reconcile kustomization book-app -n flux-system --with-source || true
kubectl -n "$APP_NS" rollout status deployment/podinfo --timeout=5m
kubectl -n "$APP_NS" get pods -l app=podinfo   -o jsonpath='{range .items[*]}
{.metadata.name}{"  "}{.spec.containers[0].image}{" "}
{.status.containerStatuses[0].imageID}{"
"}{end}'   | tee evidence/runtime-release.txt
```

通过同一 Service 契约验证新的响应，并保存控制器修订版。

```
kubectl -n "$APP_NS" port-forward service/podinfo 9898:9898 &
PF_PID=$!
sleep 3
curl -fsS http://127.0.0.1:9898/version   | tee evidence/version-release.json
kill "$PF_PID"
kubectl -n flux-system get gitrepository,kustomization -o yaml   >
evidence/flux-after-release.yaml
```

12.15.6 证明漂移修正

把副本数从二更改为一，引入一次无害的命令式漂移。该命令不是发布，而是一次协调测试。由于 Git 声明拥有该字段，Flux 应恢复两个副本。

```
kubectl -n "$APP_NS" scale deployment/podinfo --replicas=1
kubectl -n "$APP_NS" get deployment/podinfo   -o jsonpath='{.spec.replicas}{"
"}'   | tee evidence/replicas-immediate.txt
```

```
sleep 90
kubectl -n "$APP_NS" get deployment/podinfo   -o jsonpath='{.spec.replicas}{"
"}'   | tee evidence/replicas-reconciled.txt
```

如果值仍为一，请检查 Kustomization 间隔、源修订版、控制器事件，以及副本是否由另一个控制器拥有。该漂移测试中不要启用 HPA，否则 HPA 与 Git 会争夺同一字段。

12.15.7 通过 Git Revert 回滚

为发布提交创建一次 revert，并通过相同的治理路径合并。这样会产生一个恢复先前期望状态的新提交，同时保留完整历史。它优于重置或删除历史。

```
git checkout main
git pull --ff-only
git log --oneline -5
# Replace RELEASE_COMMIT with the commit that promoted 6.14.1.
git revert RELEASE_COMMIT
git push origin main
git rev-parse HEAD | tee evidence/rollback-config-sha.txt

flux reconcile source git book-gitops -n flux-system --with-source || true
flux reconcile kustomization book-app -n flux-system --with-source || true
kubectl -n "$APP_NS" rollout status deployment/podinfo --timeout=5m
kubectl -n "$APP_NS" get pods -l app=podinfo   -o jsonpath='{range .items[*]}
{.metadata.name}{"  "}{.spec.containers[0].image}{" "}
{.status.containerStatuses[0].imageID}{"
"}{end}'   | tee evidence/runtime-rollback.txt
```

验证应用报告已恢复的版本，并且 Flux 把回滚提交报告为已应用源修订版。

```
kubectl -n "$APP_NS" port-forward service/podinfo 9898:9898 &
PF_PID=$!
sleep 3
curl -fsS http://127.0.0.1:9898/version   | tee evidence/version-rollback.json
kill "$PF_PID"
kubectl -n flux-system get gitrepository,kustomization -o yaml   >
evidence/flux-after-rollback.yaml
```

12.15.8 证据包

表 12-17 实验 12 证据包。

证据	验收标准
存储库历史	初始、发布和还原提交均存在；发布差异只更改期望状态。

证据	验收标准
Flux 配置	Azure fluxConfiguration 资源以及集群内 GitRepository/Kustomization 均为 Ready。
初始状态	Podinfo 6.14.0 期望引用、运行时 imageID 摘要、发布成功和 /version 响应。
发布状态	Podinfo 6.14.1 期望引用、新的运行时 imageID 摘要、已应用源修订版和 /version 响应。
漂移测试	命令式副本变更被修正回已声明数量。
回滚状态	revert 提交成为已应用修订版，运行时恢复到 6.14.0。
没有直接权威变更	发布或持久回滚没有使用 kubectl apply/set image/edit 命令。
清理	移除 Flux 配置与命名空间；如果集群/资源组仅为实验创建，则将其删除。

为证据文件创建清单并计算哈希。在生产系统中，应把证据包存储在不可变或访问受控的证据存储库中，并设定保留规则。

```
find evidence -type f -maxdepth 1 -print0   | sort -z   | xargs -0 shasum -a 256
> evidence/SHA256SUMS.txt
```

12.15.9 清理与零残留验证

删除 Azure Flux 配置。如果其他配置仍在使用该扩展，扩展可以保留。删除应用命名空间。如果集群与资源组只为本实验创建，则删除资源组并验证完成。

```
az k8s-configuration flux delete   --resource-group "$RG"   --cluster-name
"$AKS"   --cluster-type managedClusters   --name "$FLUX_CONFIG"   --yes

kubectl delete namespace "$APP_NS" --ignore-not-found
az k8s-configuration flux list   --resource-group "$RG"   --cluster-name "$AKS"
--cluster-type managedClusters   --output table
kubectl get namespace "$APP_NS" 2>&1   | tee evidence/namespace-after-
cleanup.txt
```

如果存储库只为实验存在，请在保留证据包后将其归档或删除。如果删除临时 AKS 资源组，请记录删除操作，并验证另一个资源组中没有残留托管磁盘、公网 IP、负载均衡器或监控资源。

12.15.10 故障诊断

表 12-18 实验 12 常见故障与诊断证据。

症状	可能原因	证据与响应
Flux 配置不是 Ready	存储库 URL/分支/路径错误、扩展问题、网络出口或源身份验证。	检查 Azure fluxConfiguration 状态、GitRepository 条件和控制器日志。
Kustomization Ready=False	YAML 无效、缺少命名空间、健康检查失败或策略拒绝。	检查 Kustomization 事件、kubectl describe、准入消息和渲染后的清单。
ImagePullBackOff	标签/摘要不存在、注册表不可用或拉取授权失败。	检查 Pod 事件与镜像引用；确认公开或需要凭据的注册表访问。
发布超过截止时间	就绪失败、配额/容量、资源不足或应用错误。	使用发布状态、Deployment 条件、Pod 事件和日志进行检查。
漂移未被修正	协调已暂停、间隔尚未到达、源陈旧，或字段被忽略/由其他对象拥有。	检查暂停状态、源修订版、Kustomization 间隔和托管字段。
Git revert 已合并，但运行时未变化	控制器尚未获取新修订版，或应用失败。	比较 Git HEAD、GitRepository 制品修订版、Kustomization 最后应用修订版和事件。
运行时撤销被覆盖	Git 仍声明较新版本，并且自愈处于活动状态。	这是预期的 GitOps 行为；还原或更新权威期望状态。

12.15.11 多云转译

可以转译实验的工程意图，而无需声称托管服务完全相同。在阿里云上，ACK One GitOps 可以托管 Argo CD，并跨 ACK 集群协调存储库。在腾讯云 TKE 或火山引擎 VKE 上，可以用自主管理的 Flux 或 Argo CD 执行相同的拉取/协调/漂移实验，或将结果与相应流水线驱动部署服务比较，并记录不同的凭据和漂移模型。验收标准保持不变：版本化期望状态、由控制器应用发布、证据、以源为权威的回滚，以及零残留。

12.16 CloudDesk 能力交接

CloudDesk FinOps Agent 不应获得不受限的集群凭据或通用“运行 kubectl”工具。它应通过保留治理路径的狭窄能力与交付控制平面交互。

表 12-19 由本章派生的 CloudDesk 交付能力。

能力	行为	风险与审批边界
gitops.change.propose	创建一个分支与拉取请求，更改有界期望状态路径；附加制品和证据身份。	不合并，也不变更集群。要求存储库路径允许列表、架构验证与所有者审查。

能力	行为	风险与审批边界
gitops.sync.observe	从控制器和运行时证据读取源修订版、协调条件、漂移和已部署主体。	只读。不得泄露无关租户或命名空间状态。
gitops.rollback.trigger	为具名发布准备还原拉取请求，并解释兼容性约束。	需要批准；不能强制推送，也不能悄然执行运行时撤销。
delivery.release.evidence	把源、制品、配置、批准、发布、遥测和成本记录汇总成签名证据包。	按保留与访问策略执行只读收集；必须报告缺失链接。

这些工具强化了第二十章确立的分离：智能体可以提出并解释变更，但存储库规则、策略门禁、批准者、协调器和可观测性系统仍保有权限。智能体绝不能仅仅因为云 API 允许，就把优化建议转化为未经审查的生产变更。

12.17 要点

- 持续交付是从意图到已验证结果的受管理反馈回路，而不只是构建与部署脚本。
- 制品只构建一次，以不可变方式标识，并通过更改版本化期望状态进行提升。
- 环境是策略、身份、数据、网络、SLO、配额、成本与证据边界，而不仅是分支或命名空间。
- OpenGitOps 要求声明式期望状态、版本化不可变历史、自动拉取与持续协调。
- 拉取式 GitOps 收窄外部部署凭据，却赋予协调器高价值权限，仍需最小权限与多租户控制。
- Git 历史本身不足以治理；拉取请求、所有权、必需检查、审批、策略、例外与决策记录共同形成控制契约。
- 必须对漂移进行分类。未经授权的人工变更、控制器拥有的字段、生成的机密、准入变更和紧急干预需要不同响应。
- 渐进式交付使用流量与证据控制影响范围。技术健康、安全、业务价值、AI 质量与成本都可以成为提升门禁。
- 运行时撤销、源还原与向前修复是不同的恢复机制。只有权威期望状态发生变更，GitOps 回滚才能持久。
- 发布证据应连接源、制品、配置、批准、控制器状态、运行时身份、遥测、成本和结果。
- AI 原生交付必须把模型、提示词、知识、工具、安全策略、评估和智能体定义版本化为受治理的发布主体。
- 多云交付应保留工程意图，同时记录凭据、协调、集群舰队管理、证据和责任方面的不等价之处。

参考资料与延伸阅读

OpenGitOps。“GitOps 原则”，1.0.0 版。https://opengitops.dev/。

Flux 项目。“核心概念”。https://fluxcd.io/flux/concepts/。

Flux 项目。“Kustomization”。https://fluxcd.io/flux/components/kustomize/kustomizations/。

Flux 项目。“监控与警报”。https://fluxcd.io/flux/monitoring/alerts/。

Microsoft。“使用 GitOps（Flux v2）部署应用”。Azure Arc-enabled Kubernetes 与 AKS 文档。

Microsoft。“Azure Kubernetes 服务的 GitOps”。Azure 架构中心。

Microsoft。az k8s-configuration flux 与 Flux Kustomization 的 Azure CLI 参考。

Microsoft。Microsoft.KubernetesConfiguration/fluxConfigurations 的 ARM/Bicep 参考。

Microsoft。“将 Azure Login Action 与 OpenID Connect 配合使用”。适用于 Azure 的 GitHub Actions 文档。

Argo 项目。“Argo CD：用于 Kubernetes 的声明式 GitOps 持续交付”。https://argo-cd.readthedocs.io/。

Argo 项目。“自动同步策略”。Argo CD 文档。

Argo 项目。“Argo Rollouts：Kubernetes 渐进式交付控制器”。https://argoproj.github.io/argo-rollouts/。

Kubernetes 文档。“Deployment”“无停机更新 Deployment”及 kubectl rollout 参考。

GitHub 文档。“规则集的可用规则”“关于代码所有者”和“使用环境进行部署”。

GitHub 文档。“OpenID Connect”和“制品证明”。

DORA。“DORA 软件交付绩效指标”。https://dora.dev/guides/dora-metrics/。

CNCF。《云原生安全白皮书》以及 Flux 与 Argo 项目文档。

SLSA。“软件制品供应链级别，1.0 版”。https://slsa.dev/。

in-toto 项目。“保护软件供应链完整性的框架”。https://in-toto.io/。

OpenSSF。“安全软件开发基础”及相关供应链指南。

Alibaba Cloud。“ACK One GitOps 概述”“使用 GitOps”以及“使用 Argo Rollouts 实现金丝雀发布”。

Tencent Cloud。“CODING 持续部署”以及部署到 Kubernetes/TKE 的文档。

Volcengine。面向 VKE 与 Kubernetes 部署的持续交付/CodePipeline 文档。

Stefan Prodan。“Podinfo”存储库与不可变发布历史。https://github.com/stefanprodan/podinfo。

Forsgren, Nicole；Humble, Jez；Kim, Gene。《Accelerate：精益软件与 DevOps 科学》。IT Revolution，2018。

Beyer, Betsy；Jones, Chris；Petoff, Jennifer；Murphy, Niall Richard，编。《站点可靠性工程》。O’Reilly Media，2016。

Humble, Jez；Farley, David。《持续交付》。Addison-Wesley，2010。

Burns, Brendan；Beda, Joe；Hightower, Kelsey；Evenson, Lachlan。《Kubernetes：权威指南》第 3 版。O’Reilly Media，2022。

运维与价值控制平面

系统投入运行之后，谁知道服务是否可靠、费用如何产生、策略是否有效？从一次可观察的运行结果出发，将可靠性、支出和安全态势连接到有责任人、可追踪的变更。

13 作为托管反馈回路的可观测性

14 作为云控制平面的 FinOps

15 安全态势、合规与组织治理

第四篇 / 第 13 章

作为托管反馈回路的可观测性

从关联遥测到 SLO 引导的决策、持续验证与韧性证据

日志、指标、追踪和剖析只是原始信号。只有当云控制平面把这些信号转化为有关发布风险、服务可靠性、成本和恢复的可解释决策时，可观测性才成为一种工程能力。

本章内容

- 为何可观测性是一种托管反馈回路，而不是遥测收集项目
- 为何日志、指标、追踪、剖析和事件是信号，而非彼此独立的支柱
- OpenTelemetry 如何建立可移植的插桩与上下文传播契约
- Azure Monitor 如何将收集、存储、分析、告警和响应拆分为可治理的资源
- SLI、SLO、错误预算和消耗率如何把用户体验转化为运行策略
- 告警规则、操作组和处理规则如何路由决策，而非仅仅发送通知
- AIOps 可以在哪些方面辅助异常检测、关联和调查，以及哪些方面必须保留人的权限
- 持续验证如何将发布与功能变更同可观测证据连接起来
- 混沌实验如何将可观测性作为韧性工程的判定依据
- 保留期、采样、基数、隐私和摄取量如何使可观测性成为一项 FinOps 议题
- Azure、阿里云、腾讯云和火山引擎如何以不同的控制平面对象表达相同的可观测性意图
- 如何为服务插桩、定义 SLO、创建可验证告警、收集证据并清理资源

即使仪表板上的图表全部呈绿色，用户也可能无法完成真正重要的任务。CPU 利用率可能很低，是因为请求在到达应用之前就被拒绝了。平均延迟看起来可能很健康，但少量高价值租户却在经历超时。模型端点可能返回 HTTP 200，却没有给出有依据的答案。队列看似稳定，可能只是因为消息被丢弃而非处理。上述每种情况下都有监控数据，但系统并未通过证据得到管理。

本章采用结果优先的定义：可观测性是利用关联遥测和运行上下文，对系统行为提出并回答问题的能力。当控制平面规定数据从何处收集、谁可以查询、保留多久、由哪些服务目标治理决策、采用何种告警策略路由响应，以及如何验证干预效果时，可观测性就成为一种托管云能力。

核心反馈回路说起来简单，实施起来要求很高：系统行为产生遥测；根据服务目标解释遥测；策略选择操作；操作改变系统；后续遥测再验证预期改进是否发生。只有证据真正改变了决策，回路才算闭合。

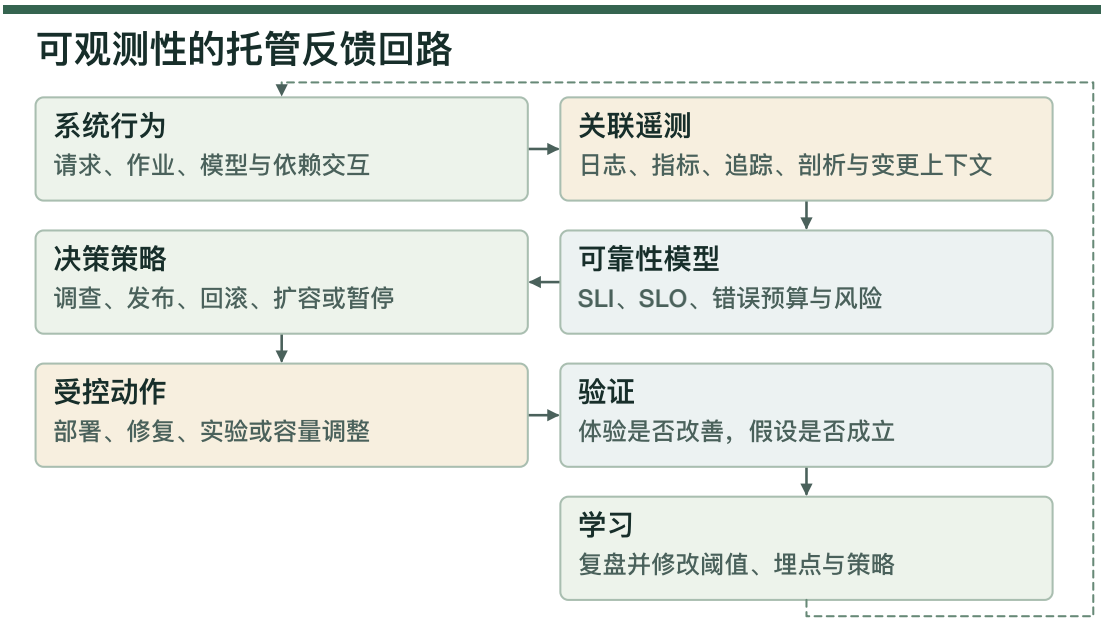


图 13-1 托管可观测性反馈回路。只有当信号能够支持可靠性模型、决策策略、受控操作、验证和学习时，它们才真正有用。

13.1 可观测性是一套控制系统，而非数据收集项目

“监控”一词通常指检查已知条件：虚拟机是否正在运行？CPU 利用率是否超过阈值？HTTP 端点是否返回了响应？这些问题仍然有用，但对于分布式和 AI 原生系统远远不够。现代故障经常由复杂交互涌现：没有任何单一组件明显损坏，但特定版本、区域、租户、功能开关、下游依赖和请求形态的组合却产生了不可接受的行为。

可观测性支持调查已知与新型故障模式。它使运维人员或开发者能够从用户可见的症状出发，定位到可以解释该症状的具体路径、依赖、配置、发布版本或数据条件。云控制平面则加入大规模运行这项能力所需的资源与策略：工作区、收集规则、诊断设置、身份、私有端点、保留计划、告警规则、操作组、工作簿和自动化目标。

控制系统包含测量状态、期望状态、比较二者的策略，以及改变系统的执行器。可观测性提供测量状态和评估变更所需的证据。SLO 描述期望的服务结果。告警、发布门禁、自动伸缩、回滚和补救都是执行器或决策路径。持续验证用于检验操作是否使服务朝目标方向推进。

表 13-1 监控、遥测、可观测性与运行控制

概念	首要问题	典型工件	单独使用时的失效方式
监控	某个已知条件是否超出了预期范围？	仪表板、运行状况检查、阈值告警	会遗漏新型故障模式，并且可能只优化组件健康，而非用户结果。
遥测	系统发出了哪些信号？	日志、指标、追踪、剖析、事件	产生大量数据，却不保证关联、解释或操作。
可观测性	为何出现这种行为？哪些人受到影响？	关联查询、追踪、维度、变更上下文	如果不与目标和策略相连接，它可能仍只是一个调查工具。
托管反馈回路	应做出什么决策、由哪些证据授权，以及决策是否奏效？	SLI/SLO、告警规则、操作路径、实验报告、审计轨迹	当目标、权限、证据质量或验证含糊不清时就会失效。

目标并非集中保存每一个字节的运行数据，而是建立从用户结果到工程决策的可信证据路径。这条路径应当可查询、可解释、有权限控制、具有成本意识且可测试。

13.1.1 控制平面本身也有可观测资源

可观测性平台不是一扇被动窗口，而是具有资源和生命周期的托管系统。工作区有区域、保留策略、访问模型、摄取端点、配额和成本。数据收集规则选择来源和转换。告警规则包含查询、评估窗口、严重性、作用域和响应路径。混沌工作区或实验则包含身份、目标作用域、权限、操作、护栏和证据。

这意味着，可观测性设计必须回答本书其他章节中同样的六个控制平面问题：资源是什么？谁可以创建、更改、查询或删除它？数据通过哪条网络路径摄取和查询？如何以代码方式部署配置？会产生哪些审计证据？如何对服务进行计量、限制和优化？

注意：即使生产数据库受到了正确保护，查询权限过宽的监控工作区仍可能暴露敏感的用户、租户或安全数据。遥测是需要治理的数据，而非无害的废气。

13.2 信号是原材料，而非支柱

“可观测性的三大支柱”这一常见说法可用作入门助记符，但也可能造成三个运行孤岛。团队分别购买一套日志系统、指标系统和追踪系统，最后却发现无法跨系统跟踪同一个请求。更可靠的模型是把日志、指标、追踪、剖析和事件视为共同运行故事的不同投影。

各种信号仍不相同，因为它们的语义和经济性不同。指标能够高效聚合，追踪保留因果结构，日志保留事件细节，剖析揭示代码在哪里消耗资源，变更事件则说明部署、策略分配、功能开关或配置修订何时改变了系统。统一并不意味着把每种信号强行装入同一种存储格式，而是携带共同上下文，使调查可以在不同信号之间移动而不丢失身份。



图 13-2 日志、指标、追踪、剖析和事件保留各自优势；当它们通过追踪、服务、资源、部署和业务上下文相关联时，用处会显著提高。

13.2.1 日志：结构化故事，而非文本文件

生产日志应当是带字段的事件，而非仅供人工阅读、长度无界的句子。日志至少应包含时间戳、严重性、服务名称、环境、版本、操作、结果和关联标识符。适当时还可以包含租户、区域、功能开关、模型部署、提示模板版本或业务操作。敏感值应予以编校，或以稳定标识符表示，而不应复制到事件中。

工程师需要事件细节时，日志最有价值，例如异常、策略决策、被拒绝的工具调用、重试原因、架构不匹配，或工作流选择的具体分支。如果每个请求都反复输出同一批非结构化上下文、嵌入机密或个人数据，或无限期保留高容量调试输出，日志的成本会上升而效用会下降。

13.2.2 指标：用于趋势和策略的压缩视图

指标是与维度和时间关联的数值观测。计数器、仪表、直方图和摘要可以紧凑表示大量活动。指标非常适合表示速率、比率、分布、容量趋势、自动伸缩信号和 SLO 计算。但如果允许维度无限增长，它们也很危险。用户 ID、请求 ID、文档 ID 或提示文本通常不应成为指标标签。

CPU、内存、I/O、队列深度和令牌吞吐量等平台指标都是有用的诊断输入，却很少是最终可靠性目标。用户关心的是：请求能否在可接受时间内成功完成，生成的答案是否有依据，批处理是否在截止时间前完成，或者财务建议是否基于最新成本数据生成。

13.2.3 追踪：因果与时间骨架

分布式追踪把一次端到端操作表示为跨度组成的树或图。每个跨度记录命名操作、持续时间、状态、属性、事件和关系。上下文传播把追踪身份跨进程和网络边界传递，使入站请求、数据库查询、消息发布、队列消费、模型调用、检索查询和工具执行能够组装成一条

因果路径。

追踪对 AI 原生应用尤为重要，因为单个用户请求经常扩展为多次检索操作、模型调用、安全检查、工具执行和异步工作项。即使每个独立组件从聚合角度看都很健康，响应仍可能很慢。追踪会揭示关键路径，以及延迟在各编排步骤之间的分布。

13.2.4 剖析、事件和业务上下文

持续剖析增加了有关 CPU 时间、分配、阻塞或其他资源热点的代码级证据。它补充而非取代追踪：追踪显示哪个操作缓慢，剖析则显示哪个代码路径消耗了时间。部署、配置、策略和功能开关事件会增加变更上下文。业务维度把技术行为连接到经历该行为的人群。

表 13-2 信号优势、典型问题和常见误用

信号	最适合回答的问题	控制平面关注点	常见误用
日志	究竟发生了什么？发生了什么决策或异常？	架构、编校、保留、查询访问、摄取量	非结构化文本、机密、过量调试日志、缺少关联 ID。
指标	某个速率、比率、分布或容量趋势是否正在改变？	标签策略、聚合、保留、告警评估、基数	把主机平均值用作用户 SLO；标签包含无界标识符。
追踪	时间或故障在请求路径的哪些位置传播？	采样、上下文传播、租户边界、依赖插桩	采样丢失罕见故障；把每个跨度属性都视为无害。
剖析	哪些代码路径消耗 CPU、内存或等待时间？	收集开销、符号访问、保留、隐私	没有明确问题或成本边界却持续收集。
事件与变更	症状之前发生了什么部署、策略、功能或拓扑变更？	审计集成、时钟对齐、不可变标识符	发布数据与应用遥测彼此分离。
业务上下文	哪些客户、租户、工作流或结果受到影响？	数据分类、假名化、访问作用域	把敏感业务数据复制到标签或日志中。

13.3 OpenTelemetry 作为插桩契约

OpenTelemetry 提供供应商中立的 API、SDK、语义约定、上下文传播，以及用于接收、处理和导出遥测的 Collector。其主要架构价值并不在于使每个后端完全等价，而在于让应用代码以标准方式描述操作，同时由部署团队选择在何处处理和存储遥测。

插桩契约分为四层。应用和库插桩创建跨度、指标与日志；上下文传播携带因果身份；Collector 通过管道接收并处理数据；Exporter 把数据发送到一个或多个后端。后端仍决定查询语言、保留、分析、告警、访问控制和成本。

表 13-3 OpenTelemetry 架构中的责任

层	责任	设计问题
API 和 SDK	创建遥测并附加语义属性。	哪些操作很重要？哪些属性稳定、有界且安全？
插桩库	观测框架、HTTP、数据库、消息传递、运行时和 AI 库。	哪些内容是自动的？哪些位置需要手动跨度或业务指标？
上下文传播	跨进程边界携带追踪和 baggage 上下文。	哪些协议会保留上下文？哪些内容绝不能放入 baggage？
Collector	接收、批处理、筛选、采样、转换、扩充、路由和导出。	它在哪里运行？如何伸缩、保护和监控？
后端	存储、查询、可视化、关联、告警和保留。	访问模型、查询语言、保留、成本和证据保证是什么？

13.3.1 上下文传播是连接组织

只有每一跳都保留追踪标识符时，它才有价值。HTTP 标头、消息元数据、后台作业和工具调用都必须传播上下文。在支持时，日志应包含追踪和跨度标识符。指标可以通过样本关联，也可以从追踪中导出。服务、版本、资源、环境和部署属性提供了比较变更前后行为所需的稳定维度。

OpenTelemetry baggage 可以跨边界携带上下文值，但应谨慎使用。Baggage 会传播到下游服务，并可能跨越信任边界。敏感信息、令牌、原始提示、客户名称或不受限制的租户数据不应放在其中。稳定的租户标识符在受控环境中可能有用，但必须审查接收服务、Exporter 和存储访问模型。

13.3.2 供应商中立的插桩不等于无供应商的运维

OpenTelemetry 降低了插桩边界的耦合，但各云平台的运行语义仍然不同。某项服务可能把应用追踪存放在日志分析工作区，另一项放入专用 APM 存储，还有一项存入日志服务。查询语言、采样选项、保留层级、指标限制、告警语义和计费单位都不同。因此，可移植性意味着保留意图和证据模型，而不是假定产品可以互换。

13.3.3 采样是一项证据策略

采样通过仅保留部分追踪总体来控制数据量。头部采样在追踪开始附近作出决定，运行上很简单，却无法知道追踪后来是否会变慢或失败。尾部采样在获得更多追踪数据后再决定，因此可以制定保留错误、高延迟、重要租户或异常路径的策略。尾部采样需要缓冲，并要求 Collector 架构能够看到完整追踪。

必须根据系统需要回答的问题来评估采样。1% 的随机样本也许足够判断总体延迟趋势，却可能抹去罕见的跨租户故障或授权故障。SLO 计算通常应依赖完整或经过适当聚合的请求计数，而非采样后的追踪。采样配置本身应当有版本并且可观测。

注意：遥测管道需要自己的服务目标。如果跨度迟到、上下文断裂，或者收集恰好在过载期间丢失数据，可观测性系统就无法充当可靠的判定依据。

13.4 Azure 作为参考可观测性控制平面

本章以 Azure Monitor 为参考平台，因为它把主要可观测性责任公开为云资源。Application Insights 提供应用性能监控和分布式追踪。Log Analytics 工作区使用 Kusto 查询语言（KQL）存储和查询日志与追踪数据。Azure Monitor 工作区目前为 Prometheus 指标提供托管存储库，并公开 PromQL。平台指标仍通过 Azure Monitor Metrics 提供。Workbooks 和 Azure Managed Grafana 提供可视化。告警规则、操作组和告警处理规则负责路由响应。诊断设置、Azure Monitor Agent、数据收集规则、数据收集端点和 OpenTelemetry 集成负责治理收集。

该架构刻意保持多元。日志和追踪与 Prometheus 指标并不具有相同的存储或查询语义。平台指标、资源日志、应用遥测和活动日志来自不同来源。管理目标是通过稳定的资源和操作上下文把它们连接起来，而不是强行塞入同一个无差别数据库。



图 13-3 Azure Monitor 参考架构。收集、存储、分析、决策路由和跨领域治理是彼此独立但又相互连接的控制平面责任。

13.4.1 工作区是治理边界

工作区不只是图表获取数据的地方。它定义区域数据边界、权限、保留行为、查询作用域和成本分配点。Log Analytics 工作区存放日志和追踪表。基于工作区的 Application Insights 资源把应用遥测写入关联的 Log Analytics 工作区。Azure Monitor 工作区保存

Prometheus 指标，并有自己的权限和容量考量。

工作区拓扑应根据访问、驻留、保留、查询和组织要求设计。每个应用一个工作区会造成碎片化；整个企业共用一个工作区又会造成访问范围过宽和成本归因噪声过多。正确边界应遵循治理和运维要求，而非普适公式。

13.4.2 收集是策略，而非安装代理

Azure Monitor Agent 和数据收集规则把数据源配置与单台机器分离。诊断设置路由受支持的资源日志和指标。数据收集端点在需要时提供明确的摄取路径。OpenTelemetry 插桩发出应用遥测。这些资源应随 IaC 一起部署，并像其他生产配置一样接受审查。

收集规则应说明收集什么、来自哪个作用域、进行哪些转换以及送往哪个目的地。转换可以在存储前删除不必要字段或规范化记录。诊断设置不应不加区分地永久启用每个类别。更多数据会增加成本，并可能拖慢调查，却不一定提高决策质量。

13.4.3 私有连接与身份

遥测经常包含资源标识符、用户操作、异常细节、请求属性和安全证据。Azure Monitor Private Link Scope 及相关私有端点可对受支持资源的摄取和查询路径施加限制。RBAC 控制谁可以配置收集、查询工作区、修改告警或执行响应自动化。在支持的情况下，代理、Collector、自动化和混沌工作区应优先使用托管身份。

13.4.4 查询语言反映信号语义

KQL 用于大规模事件与日志分析，PromQL 用于带标签的时间序列。Metrics Explorer 提供交互式指标分析。Transaction Search 和 Application Map 利用追踪与依赖关系。成熟的运行模型会教会工程师何时使用每个界面，以及如何在它们之间传递稳定标识符。

表 13-4 Azure 可观测性对象及其控制平面责任

对象	首要责任	证据或查询界面
Application Insights 资源	应用请求、依赖、异常、可用性、分布式追踪、应用地图。	KQL、事务搜索、故障和性能视图。
Log Analytics 工作区	日志和追踪存储、KQL 查询、表计划、保留、访问与导出。	KQL 查询结果、工作区审计与使用记录。
Azure Monitor 工作区	Prometheus 指标及相关配置的托管存储。	PromQL、托管 Grafana、记录与告警规则。
数据收集规则/端点	选择、转换、路由和安全接收受支持的遥测。	资源配置、摄取健康状况和目的地证据。
诊断设置	把 Azure 资源日志和受支持指标路由到选定目的地。	活动日志和目的地记录。

对象	首要责任	证据或查询界面
告警规则	评估指标、日志查询、活动事件、Prometheus 规则或其他条件。	告警实例、评估历史、条件状态。
操作组	把通知或自动化路由到可复用的目的地。	操作历史和通用告警有效负载。
告警处理规则	按作用域、筛选器和计划抑制或添加操作组。	处理规则状态和产生的告警历史。
Workbook / Managed Grafana	组合运行视图和调查路径。	已保存的查询、面板、参数、链接和仪表板版本。
Chaos Studio 工作区/场景	发现资源并运行受控韧性测试。	场景运行与报告、身份、作用域和操作历史。

13.5 SLI、SLO 与错误预算

服务级别指标衡量一种结果，服务级别目标为该指标在定义总体和时间窗口内设定可接受水平，错误预算则表示可以容忍的不可靠性。三者共同把抽象的可靠性愿望转化为可衡量的运行契约。

可用性 SLI 可以表示为符合条件事件中良好事件的比率。延迟 SLI 可以统计在阈值以内完成的合格请求数。新鲜度 SLI 可以衡量数据是否在截止时间内可用。对于 AI 应用，指标可包括成功生成有依据的答案、工具完成、符合策略的操作或检索新鲜度。指标必须代表用户或依赖系统的实际体验。

```
availability_SLI = good_eligible_requests / all_eligible_requests

latency_SLI = requests_below_latency_threshold / all_eligible_requests

error_budget_fraction = 1 - SLO_target

remaining_budget = allowed_bad_events - observed_bad_events
```



图 13-4 SLI 衡量结果，SLO 定义目标，错误预算量化可容忍的不可靠性，策略则决定响应。

13.5.1 定义总体和窗口

如果不定义符合条件的总体，比率就毫无意义。运行状况探测、负载测试、恶意流量、已知客户端误用和管理操作可能需要分别处理。排除条件必须在事件发生前记录，而不是等服务未达目标后临时编造。时间窗口应与业务和运行行为相匹配。滚动窗口提供持续压力；日历窗口与报告对齐，但会突然重置。

13.5.2 从用户结果开始，再映射到遥测

平台指标可以解释 SLO 违规，但不应自动成为 SLO。CPU 饱和可能导致延迟，但用户体验到的是延迟；队列深度可能预示错过截止时间，但用户体验到的是结果迟到。可靠性模型应从结果出发，然后识别计算、解释和预测该结果所需的遥测。

表 13-5 诊断指标与以用户为中心的 SLI

系统视角	有用的诊断信号	更可靠的以用户为中心的指标
Web API	CPU、内存、线程池、连接池	符合条件请求的成功率和延迟达标率。
事件管道	队列深度、消费者数量、重试次数	在业务截止时间前正确处理的事件百分比。
RAG 服务	向量查询延迟、top-k 大小、模型令牌速率	在响应截止时间内、带有所需引文来回答合格问题的百分比。
智能体工具执行	工作进程利用率、函数持续时间、依赖延迟	获批操作以正确、幂等方式并在策略规定时间内完成的百分比。
成本摄取	文件数、解析器 CPU、API 调用速率	在报告截止点之前摄取且可查询的必需账单记录百分比。

13.5.3 消耗率把预算消耗转化为紧迫度

一项服务可能仍满足 30 天 SLO，但当前正在快速失效。消耗率把当前坏事件发生率与在整个窗口内均匀耗尽全部错误预算的速率进行比较。消耗率大于一表示预算消耗快于计划。多窗口、多消耗率告警使用快速窗口捕获严重事件、慢速窗口识别持续劣化，既减少响应延迟，也减少瞬时噪声。

错误预算不是惩罚机制，而是为平衡变更和可靠性建立协商规则。预算健康时团队可以发布；消耗加快时则调查或降低风险；预算耗尽时，策略可以暂停非必要变更并优先处理可靠性工作。具体策略必须由产品、开发、运维和业务利益相关者共同商定。

13.6 告警即决策路由

只有当告警识别出需要明确决策的条件时，它才有用。针对每个指标阈值都呼叫值班人员会制造噪声，并使响应者习惯于忽略系统。好的告警应说明受影响的目标、作用域、证据窗口、严重性、可能的用户影响以及预期第一步操作。它应路由给同时承担责任并拥有权限的团队或自动化。

Azure Monitor 支持指标告警、日志搜索告警、活动日志告警、Prometheus 告警、服务运行状况告警及其他类型。告警规则定义受监控作用域和条件；操作组提供可复用的通知和自动化目的地；告警处理规则可以按作用域、筛选器和计划添加或抑制操作组，这对计划内维护或组织范围的路由策略很有用。

通用告警架构帮助 Webhook 和自动化消费者处理一致的信封。自定义属性可以携带服务、运行手册、SLO、所有者或部署上下文。这些属性应引用受治理的标识符，而不应把敏感遥测复制到每条通知中。

表 13-6 从症状到问责操作的告警设计

设计要素	问题	证据
目标	哪个用户或系统结果正面临风险？	SLI、SLO ID、错误预算状态。
作用域	哪些资源、服务、区域、版本或租户受到影响？	资源 ID 和有界维度。
条件	哪个查询或指标证明了问题？	有版本的查询、阈值、评估频率、窗口。
紧迫度	损害或预算消耗正在以多快速度增加？	严重性、消耗率、最低流量和持续时间。
所有者	谁承担责任并有权采取行动？	服务所有权和值班路径。
第一步操作	告警触发时应当发生什么？	运行手册、回滚门禁、容量操作、调查查询。

设计要素	问题	证据
验证	如何证明问题已经解决?	已解决条件、用户结果恢复、告警历史和操作后查询。

13.6.1 告警质量可以衡量

团队应衡量告警精度、确认时间、获得有效诊断所需时间、重复数量、具有可执行运行手册的比例，以及在未造成用户影响前解决的比例。低流量服务需要额外谨慎，因为少量错误就可能使比率不稳定。这时可能需要最低事件数、合成流量或更长窗口。这些是建模选择，而不是忽视可靠性的借口。

13.7 AIOps 是辅助分析，而非自治权限

AIOps 将统计、机器学习以及越来越多的生成式技术应用于遥测和运行上下文。有用能力包括动态阈值、异常检测、问题分组、跨信号关联、可能根因建议、容量预测，以及面向查询或调查的自然语言辅助。这些能力可以减少人类反复进行模式识别的工作量。

辅助不等于权限。异常检测器可以识别变化，却无法在缺少策略和上下文时判断该变化在业务上是否可接受。根因建议只是一种假设，而非因果证据。生成的查询可能语法正确，却检查了错误的总体。自动补救可能放大错误诊断，或者在预期租户与资源作用域之外执行操作。

表 13-7 AIOps 能力、所需证据和护栏

能力	有用输出	所需护栏
动态阈值	按季节性或历史调整后的预期范围。	最低数据质量、有界维度、可解释基线、回退阈值。
异常检测	值得调查的候选偏差。	不要把异常等同于事件；保留查询和比较窗口。
告警分组	把相关告警汇总为一个运行问题。	保留原始告警 ID、作用域、严重性和被抑制的证据。
根因辅助	经过排序的依赖、变更、追踪或资源。	公开支持证据和置信度；要求人工或策略验证。
预测	预计的饱和、容量不足或成本增长。	跟踪预测误差，并防止单个模型触发不可逆操作。
智能体式调查	生成的查询、导航和事件摘要。	默认只读；显式规定身份、作用域、审计、编校和工具边界。
自动补救	重启、伸缩、路由、回滚或隔离。	预先批准的操作、爆炸半径限制、幂等性、回滚和操作后验证。

适当的演进顺序是：先进行人在回路的辅助，再进行受策略约束的自动化，最后才针对已充分理解的条件执行范围狭窄的自治操作。系统应保留所用证据、授权操作的身份、精确目标作用域以及验证结果。

13.8 持续验证：把交付连接到反馈回路

可观测性与变更相连接时价值最高。部署、模型修订、提示模板、检索索引、功能开关、策略分配或基础设施配置都应作为运行上下文呈现。如果没有变更标记，响应者只能看出行为发生了变化，却不知道什么发生了变化。

持续验证在暴露仍有限时评估发布。金丝雀部署或分阶段部署接收受控比例的流量，系统依据同一服务目标和选定护栏指标比较新版本与基线版本。如果证据满足策略，就扩大暴露；如果错误预算消耗过快或关键不变量失效，发布就会暂停或回滚。

表 13-8 可观测发布所需的证据

阶段	控制平面工件	验证问题
发布前	有版本的部署、SLO 定义、告警策略、预期变更、回滚目标。	我们是否知道成功和失败分别是什么样？
初始暴露	金丝雀路由、功能开关、租户群组、发布批注。	测试总体是否受控且可识别？
比较	按版本、区域、群组和依赖分组的查询。	新版本与基线是否存在实质差异？
决策	包含时间窗口、最低流量、消耗率和护栏的策略门禁。	证据是否授权推广、暂停或回滚？
决策后	部署历史、查询结果、告警历史、审计和事件说明。	另一名审阅者能否复现作出该决策的原因？

这就是可观测性驱动开发的实践形式。插桩和可靠性标准应与功能同时设计，而不是在生产故障后补加。发布工程由一次性的信心行为转变为受控实验。

13.9 韧性工程：将可观测性作为判定依据

可靠系统不能仅凭没有可见故障来证明，而应针对真实中断进行测试。混沌工程是一种严谨实验：定义稳态，提出假设，注入受控故障，观察面向用户的结果，然后学习。目标不是制造戏剧性停机，而是在失控事件暴露隐含假设之前先将其发现。



图 13-5 韧性实验从可衡量的稳态开始，以能够改变架构、运维或插桩的证据结束。

安全实验需要定义目标作用域、身份、爆炸半径、持续时间、中止条件、回滚和观察者。稳态应以 SLI 表达，而不仅是组件状态。假设可以表述为：“如果一个应用实例终止，符合条件请求的成功率仍高于 99.9%，P95 延迟仍低于 300 ms，并且没有请求被路由到未经授权的租户。”

Azure Chaos Studio 提供托管韧性测试服务。截至 2026 年 8 月，Workspaces 和 Scenarios 是较新的公开预览模型，经典实验仍然可用。Workspace 使用托管身份和 RBAC，在显式作用域内执行获准操作。Scenario 报告提供结构化证据，说明执行了什么、跳过了什么、操作耗时以及运行是否成功。由于预览状态和产品模型变化快，书稿定稿时应重新核查这些细节。

表 13-9 最小实验契约

字段	必需内容
稳态	面向用户的 SLI、查询定义、可接受范围和观测窗口。
假设	有关命名中断期间系统行为的可证伪陈述。
目标和作用域	精确的资源、区域、环境和租户边界。
身份与权限	实验使用的主体及其狭窄角色分配。
爆炸半径	受影响容量、流量、持续时间和依赖系统的上限。
中止与回滚	停止实验并恢复状态的自动或人工条件。

字段	必需内容
可观测性计划	用作判定依据的信号、查询、仪表板、告警和追踪属性。
证据包	实验定义、运行报告、遥测、决策、发现和行动项。

13.10 成本、基数、保留与隐私

可观测性会消耗存储、摄取、计算、网络和人员注意力。数据量的增长速度可能超过应用流量，因为一个请求会产生许多跨度和日志记录。目标并非最低遥测成本，而是每单位成本获得最高决策价值。因此，FinOps 与可靠性共享同一设计界面。

基数是最重要的成本和性能约束之一。以请求 ID 为标签的指标会为每个请求创建新的时间序列。包含整个提示的追踪属性会增加存储和隐私风险。反复输出大对象的调试日志可能主导摄取量。维度策略应把属性分类为稳定的有界维度、高基数调查字段、敏感数据或禁用值。

表 13-10 成本与治理杠杆

杠杆	收益	误用风险
插桩范围	避免为无法支持决策的操作生成遥测。	如果关键路径或故障未插桩，就会形成盲点。
筛选与转换	在存储前删除噪声或敏感字段。	不可逆的证据丢失；转换缺陷。
采样	降低追踪和日志数据量。	罕见或租户特定事件可能消失。
聚合/记录规则	降低查询成本，并支持快速 SLO 视图。	聚合可能掩盖分布尾部或错误总体。
保留层级	使热调查数据与长期合规或趋势需要相匹配。	保留期太短会妨碍事后分析；长期热保留则浪费成本。
存档与导出	以较低成本或在独立存储中保存证据。	访问更慢，并增加治理与检索成本。
维度策略	控制指标序列并保护敏感数据。	过于严格的策略可能删除必要的诊断上下文。
工作区与租户边界	改善访问控制、驻留和成本归因。	碎片过多会使跨服务调查变得困难。
用量配额与告警	防止摄取费用失控。	硬上限可能在导致用量激增的事件期间丢弃证据。

13.10.1 可观测性管道也需要可靠性指标

服务本身可能健康，而其遥测却延迟或不完整。运维人员应衡量 Collector 可用性、导出失败、丢弃的跨度、摄取延迟、查询新鲜度、上下文传播成功率、告警评估延迟和通知交付。这些指标决定平台在事件期间是否值得信任。

可观测性 SLO 可以规定：99.9% 的合格应用请求遥测在两分钟内可查询，99.5% 的分布式请求带有追踪上下文，或关键告警通知在六十秒内送达。这些指标不能代替应用 SLO，而是限定用于评估应用 SLO 的证据质量。

13.10.2 隐私与多租户边界

遥测绝不能成为租户之间的侧信道。日志、追踪、仪表板、已保存查询、导出、告警有效负载和 AIOps 摘要应保持与其观测系统相同的租户与数据分类边界。共享工作区可能提高运行效率，但必须配备资源上下文访问控制、查询治理、编校，以及对跨租户支持工作流的显式审查。

对于 AI 应用，默认不应记录原始提示、检索到的文档文本、模型响应、工具参数和用户文件。在足够的情况下，应记录稳定标识符、模型与策略版本、令牌数、时间、安全结果和哈希值。只有在获批的诊断策略下，并设置严格访问与保留要求后，才能捕获有效负载。

13.11 多云转换：保留工程意图

主要云平台通过不同产品公开相似的可观测性意图。Azure 将应用监控、日志、Prometheus 指标、告警、可视化和韧性测试分布在 Azure Monitor 及相关资源中。阿里云将 ARMS 应用监控和 Managed Service for OpenTelemetry 与日志服务（SLS）及 Managed Service for Prometheus 结合。腾讯云可观测平台整合产品监控、APM、CLS、TMP、Grafana 和告警管理；其混沌演练平台提供托管故障注入工作流。火山引擎提供 APMPlus、日志服务（TLS）、托管 Prometheus 和 Kubernetes 可观测性能力，并为应用监控记录了 OpenTelemetry 集成。

这些服务不能按 SKU 逐项互换。可移植单元是工程契约：已插桩的操作、传播的上下文、受治理的存储、查询、SLI、告警、操作路径、保留、成本边界和实验证据。

表 13-11 多云可观测性意图映射

工程意图	Azure	阿里云	腾讯云	火山引擎
应用追踪/APM	采用 Azure Monitor OpenTelemetry Distro 的 Application Insights。	ARMS 应用监控和 Managed Service for OpenTelemetry。	腾讯云可观测平台应用性能监控；建议采用 OpenTelemetry 集成。	采用 OpenTelemetry 集成的 APMPlus。

工程意图	Azure	阿里云	腾讯云	火山引擎
日志	Log Analytics 工作区和资源诊断设置。	日志服务（SLS），包括 ARMS 可观测数据存储集成。	日志服务（CLS）。	日志服务（TLS）。
Prometheus 指标	Azure Monitor 工作区和托管 Prometheus 服务。	Managed Service for Prometheus；PromQL 和 OpenTelemetry 指标摄取。	腾讯云 Prometheus 监控服务（TMP）。	托管 Prometheus/Kubernetes 监控能力。
可视化	Azure Workbooks 和 Azure Managed Grafana。	Managed Service for Grafana 和 ARMS 仪表板。	腾讯云 Grafana 服务和腾讯云可观测平台仪表板。	TLS、APMPlus 和云原生监控界面中的仪表板。
告警与路由	Azure Monitor 告警规则、操作组和处理规则。	ARMS 告警管理和 Prometheus 告警规则。	腾讯云可观测平台告警策略和通知模板。	产品告警与监控规则；定稿时确认当前集成。
韧性测试	Azure Chaos Studio Workspaces/Scenarios 和经典实验。	混沌工程能力与开源 ChaosBlade 生态系统；验证所选托管路径。	腾讯云智能顾问-混沌演练。	已记录的混沌工程实践和产品特定工具；验证当前托管产品。
成本/保留控制	工作区保留、表计划、采样、导出、用量和 Azure 成本管理。	基于摄取量的 ARMS 限制、SLS 保留、Prometheus 存储层级。	APM 用量分析、CLS 生命周期、TMP 保留与存档。	TLS 生命周期/索引控制和服务特定保留；验证计费单位。
可移植层	OpenTelemetry、Prometheus、服务/资源属性、SLO 文档、告警与证据契约。	意图相同；产品资源层次与计费不同。	意图相同；集成与通知渠道不同。	意图相同；英文文档深度与产品边界不同。

注意：多云章节应显式说明不等价性。产品名称、保留、计费、预览状态、支持区域和托管集成都可能变化，必须在书稿定稿前重新验证。

13.12 实验 13：为服务插桩、定义 SLO 并触发可验证告警

本实验建立一个独立的可观测性反馈回路。本地 .NET 最小 API 向基于工作区的 Application Insights 资源发出遥测。你将生成成功和失败请求，用 KQL 计算五分钟成功率，创建日志搜索告警规则，触发条件，收集证据包，最后删除 Azure 资源。

本实验聚焦一项云能力：把应用遥测转化为受治理的可靠性决策。它不需要虚拟机、Kubernetes 群集或生产应用。本地执行可以降低基础设施成本，而控制平面资源仍是真实的。

实验 13 从埋点到告警证据

应用与采集	.NET 最小 API 的正常、慢速和失败端点 → Azure Monitor OpenTelemetry → 带服务身份的日志和追踪
度量与响应	工作区型 Application Insights → KQL 查询五分钟成功率与错误预算 → 定时日志告警 → 验证触发和恢复

验证包包括 KQL 结果、告警状态、资源 ID、测试流量和清理记录。

图 13-6 实验 13 建立完整但有界的反馈回路：已插桩服务、托管遥测、SLO 查询、计划告警、证据和清理。

13.12.1 能力目标与验收标准

你将证明以下陈述：

- 服务向基于工作区的 Application Insights 资源发出带有稳定服务名称的请求遥测。
- KQL 查询计算五分钟窗口内的请求总数、失败请求数、成功率和错误率阈值。
- 计划日志搜索规则评估同一查询，并在满足条件时创建 Azure Monitor 告警。
- 证据包包含资源 ID、SLO 定义、查询、测试流量计数、查询输出、告警状态和清理验证。
- 删除资源组会移除 Application Insights 资源、Log Analytics 工作区和告警规则。

13.12.2 成本与安全边界

表 13-12 实验边界

项目	决策
云资源	一个资源组、一个 Log Analytics 工作区、一个基于工作区的 Application Insights 资源，以及一条日志搜索告警规则。
应用主机	本地开发计算机；不需要 Azure 计算资源。
预期成本类别	低。遥测摄取和日志告警评估可能产生费用；验证后删除资源组。
权限	在实验资源组中创建和删除资源、创建监控资源，以及查询 Application Insights 资源。
数据策略	仅使用合成请求。不要发送生产提示、机密、个人数据或客户标识符。
区域	使用订阅中 Log Analytics 和 Application Insights 支持的 Azure 区域。
清理单元	名称以 obs-lab- 为前缀的专用资源组。

13.12.3 先决条件

- Azure CLI 已登录到允许创建监控资源的订阅。
- .NET 8 SDK 或更高的兼容 .NET SDK。
- curl 或其他 HTTP 客户端。
- Azure CLI 2.54.0 或更高版本，用于 scheduled-query 扩展。在受支持的 CLI 配置中，首次使用时会自动安装该扩展。
- 能够设置环境变量的 shell。以下命令使用 Bash 语法。

13.12.4 创建监控资源

选择受支持的位置。资源名称包含一个短随机后缀，避免重复运行实验时发生冲突。

```
set -euo pipefail

suffix=$(openssl rand -hex 3)
location="eastus"
rg="obs-lab-${suffix}"
workspace="obs-law-${suffix}"
appinsights="obs-ai-${suffix}"
alert="slo-fast-burn-${suffix}"

az group create   --name "$rg"   --location "$location"   --tags purpose=book-
lab capability=observability owner=reader

az monitor log-analytics workspace create   --resource-group "$rg"   --
workspace-name "$workspace"   --location "$location"

workspace_id=$(az monitor log-analytics workspace show   --resource-group "$rg"
--workspace-name "$workspace"   --query id -o tsv)

az monitor app-insights component create   --app "$appinsights"   --resource-
group "$rg"   --location "$location"   --workspace "$workspace_id"   --
application-type web

app_id=$(az monitor app-insights component show   --app "$appinsights"   --
resource-group "$rg"   --query id -o tsv)

connection_string=$(az monitor app-insights component show   --app
"$appinsights"   --resource-group "$rg"   --query connectionString -o tsv)

printf 'Resource group: %s
Application Insights: %s
' "$rg" "$app_id"
```

注意：在某些 Azure CLI 安装中，Application Insights CLI 命令可能需要 application-insights 扩展。如果出现提示，请允许安装官方扩展；也可以在 Azure 门户中使用相同名称和工作区创建基于工作区的资源。

13.12.5 构建已插桩的最小 API

```
mkdir -p managed-feedback-loop-lab
cd managed-feedback-loop-lab

dotnet new web -n ManagedFeedbackLoop
cd ManagedFeedbackLoop

dotnet add package Azure.Monitor.OpenTelemetry.AspNetCore
```

用以下代码替换 Program.cs。失败端点发出结构化错误并返回 HTTP 503；慢速端点则为追踪检查提供延迟路径。

```
using Azure.Monitor.OpenTelemetry.AspNetCore;

var builder = WebApplication.CreateBuilder(args);

builder.Services
    .AddOpenTelemetry()
    .UseAzureMonitor();

var app = builder.Build();

app.MapGet("/ok", () => Results.Ok(new
{
    status = "ok",
    service = "managed-feedback-loop"
}));

app.MapGet("/slow", async () =>
{
    await Task.Delay(1500);
    return Results.Ok(new { status = "slow-but-successful" });
});

app.MapGet("/fail", (ILogger<Program> logger) =>
{
    logger.LogError(
        "Synthetic failure for observability lab. Code={Code}",
        "LAB_FAILURE");
    return Results.StatusCode(StatusCodes.Status503ServiceUnavailable);
});

app.Run();
```

在单独终端中启动应用。连接字符串是一项摄取配置值，不能替代生产身份和网络设计。

```
export APPLICATIONINSIGHTS_CONNECTION_STRING="$connection_string"
export OTEL_SERVICE_NAME="managed-feedback-loop"
export OTEL_RESOURCE_ATTRIBUTES="deployment.environment=book-
lab,service.version=1.0.0"

dotnet run --urls http://127.0.0.1:5050
```

13.12.6 生成基线流量和失败流量

在第一个终端中生成有界的合成总体。这些命令刻意产生超过十个请求，使告警规则能够采用最低数据量护栏。

```
for i in $(seq 1 50); do
  curl -s http://127.0.0.1:5050/ok > /dev/null
done

for i in $(seq 1 5); do
  curl -s http://127.0.0.1:5050/slow > /dev/null
done

for i in $(seq 1 10); do
  curl -s -o /dev/null http://127.0.0.1:5050/fail
done
```

遥测摄取是异步的。不要等待几分钟就断定插桩失败。请检查本地进程中的 Exporter 错误、确认连接字符串，并在更改设计前查询更长的时间窗口。

13.12.7 定义 SLI、SLO 和实验告警策略

本实验使用请求成功 SLI。在生产系统中，运行状况探测、测试流量、被拒绝的客户端误用及其他总体都应显式分类。对于这项合成服务，每个请求都符合条件。

表 13-13 实验可靠性模型

字段	定义
SLI	成功的合格请求数除以所有合格请求数。
SLO	在滚动 30 天窗口内达到 99.0% 的请求成功率。放宽后的目标使少量实验总体更容易推理。
错误预算	在目标窗口内允许 1.0% 的坏请求。
实验快速消耗条件	五分钟内至少有 10 个请求，且观测错误率高于 1.0%。
决策	创建严重性为 2 的告警实例。本实验不附加通知端点或自动补救。
解决证据	仅生成成功流量且评估窗口向前推进后，查询不再返回违规行。

针对 30 天 SLO 的生产告警策略应采用消耗率推理和多个窗口。实验中的简化条件是刻意保持透明的：它证明控制平面路径，而不假装五分钟比率就是完整的生产策略。

13.12.8 查询请求成功指标

Application Insights 请求遥测可通过工作区表 AppRequests 获取。在 Application Insights Logs 界面中运行以下 KQL 查询。它会对过去五分钟内的所有请求进行分组，并且仅在违反实验条件时返回一行。

```
AppRequests
| where TimeGenerated > ago(5m)
| summarize
    Total = sum(ItemCount),
    Failed = sumif(ItemCount, Success == false)
| extend ErrorRatePct =
    iff(Total == 0, 0.0, 100.0 * todouble(Failed) / todouble(Total))
| extend SuccessRatePct = 100.0 - ErrorRatePct
| where Total >= 10 and ErrorRatePct > 1.0
| project Total, Failed, SuccessRatePct, ErrorRatePct
```

记录查询文本、时间窗口、结果和 Application Insights 资源 ID。单独的屏幕截图证据较弱，因为它没有保存精确查询或总体定义。

13.12.9 创建计划日志搜索告警

把查询存入 shell 变量并创建计划查询规则。此 CLI 命令由 Azure CLI 2.54.0 或更高版本的正式版 scheduled-query 扩展提供。

```
alert_query="AppRequests
| where TimeGenerated > ago(5m)
| summarize Total=sum(ItemCount), Failed=sumif(ItemCount, Success == false)
| extend ErrorRatePct=iff(Total == 0, 0.0, 100.0 * todouble(Failed) /
todouble(Total))
| where Total >= 10 and ErrorRatePct > 1.0
| project Total, Failed, ErrorRatePct"
```

```
az monitor scheduled-query create   --resource-group "$rg" --name "$alert" --
location "$location"   --scopes "$app_id"   --condition "count
'FailureBudgetBurn' > 0"   --condition-query FailureBudgetBurn="$alert_query"
--evaluation-frequency 1m --window-size 5m   --severity 2 --auto-mitigate true
--description "Fast-burn guard for the request-success SLO lab."   --tags
capability=observability slo=request-success
```

该告警没有操作组，因此只会记录告警实例，而不会通知人员或执行自动化。这是有意设计：通知和补救需要真实的所有权、身份与安全设计。本实验先证明评估和证据。

13.12.10 触发并验证告警

规则存在后再生成一批失败请求，然后等待至少两个评估周期。

```
for i in $(seq 1 25); do
  curl -s -o /dev/null http://127.0.0.1:5050/fail
done

sleep 180
```

验证规则定义：

```
az monitor scheduled-query show   --resource-group "$rg"   --name "$alert"   --
output jsonc
```

查看 Azure Monitor > Alerts，并筛选到实验资源组。记录告警规则、监控条件、严重性、开始时间、目标资源和告警 ID。如果没有告警出现，应先重新运行 KQL 查询。未返回任何行的查询无法触发规则。不要盲目降低阈值；应验证遥测新鲜度、时间窗口、表名和最小计数。

仅生成成功流量，并等待五分钟窗口越过合成失败。启用自动缓解后，条件不再返回违规行时告警应当解决。

```
for i in $(seq 1 100); do
  curl -s http://127.0.0.1:5050/ok > /dev/null
done
```

13.12.11 组装证据包

表 13-14 必需证据

证据	记录内容
资源图	资源组、工作区、Application Insights 资源和告警规则 ID。
插桩契约	服务名称、环境和版本资源属性；使用的软件包版本。
流量总体	发送到 /ok、/slow 和 /fail 的计数及大致时间戳。
可靠性模型	SLI 分子与分母、目标、窗口、错误预算和简化的快速消耗条件。
查询证据	精确 KQL、查询时间、结果行和遥测新鲜度。

证据	记录内容
告警证据	规则 JSON、告警实例 ID、监控条件、严重性和解决状态。
负面证据	未附加操作组或自动补救。
清理证据	资源组删除命令，以及随后值为 false 的 az group exists 结果。

13.12.12 清理并验证删除

停止本地应用，返回定义实验变量的 shell，并删除专用资源组。

```
az group delete   --name "$rg"   --yes

az group exists --name "$rg"
# 预期最终值：false
```

资源组是清理单元。确认其外部未创建任何告警规则、操作组、工作区、数据收集端点或其他监控资源。已删除工作区中保留的遥测遵循 Azure 的删除与恢复行为；不要在一次性实验中使用敏感数据。

13.12.13 实验的多云转换

本实验刻意以 Azure 为完整实现，但其工程契约是可移植的。在阿里云上，使用 OpenTelemetry 为同一服务插桩，并把追踪路由到 ARMS Managed Service for OpenTelemetry 或应用监控，按需使用 SLS 和 Managed Service for Prometheus，再定义 ARMS 告警策略。在腾讯云上，使用集成 OpenTelemetry 的 APM、CLS 或 TMP，以及腾讯云可观测平台告警策略。在火山引擎上，使用集成 OpenTelemetry 的 APMPlus 和相应日志或指标服务。保持相同的服务名称、SLI 定义、时间窗口、最小总体、证据字段和清理契约。

注意：不要把控制台复选框视为等价。应确认各个平台的实际数据模型、查询语言、告警评估语义、通知路径、保留、计费单位和访问边界。

13.13 移交给 CloudDesk FinOps Agent

CloudDesk FinOps Agent 以后可以使用可观测性与可靠性能力，但本章不实现它的租户系统、Serverless 运行时、桌面审批或跨租户隔离。可复用输出是一组受治理的能力契约。

```
observability.signal.collect(source, signalPolicy, destination)
  -> resourceIds, ingestionPath, schema, retention, samplingPolicy
  证据：collectionHealth, lastSeen, droppedData, costScope
```

```
reliability.slo.evaluate(serviceId, sliDefinition, objective, window)
  -> attainedRatio, errorBudgetRemaining, burnRate, decisionState
  证据：query, population, timeWindow, sourceCompleteness, policyVersion

observability.anomaly.explain(signalScope, timeWindow, changeContext)
  -> findings, correlatedTraces, candidateChanges, confidence
  证据：queries, resourceIds, traceIds, alertIds, humanDisposition

resilience.experiment.verify(experimentId, steadyState, hypothesis)
  -> outcome, violatedGuardrails, recoveryTime, recommendations
  证据：runReport, telemetryQueries, targetScope, executingIdentity
```

未来的 Agent 默认应只获得只读查询能力。创建告警、抑制通知、执行补救或运行混沌操作，需要独立身份与策略门禁。Agent 必须引用资源 ID、查询、窗口和告警或实验记录，以便人类复现其结论。

13.14 关键要点

可观测性是托管反馈回路：系统行为产生证据，可靠性策略选择决策，后续证据验证结果。

日志、指标、追踪、剖析和变更事件是互补信号。共享上下文支持在它们之间导航，同时保留各自不同的语义与成本。

OpenTelemetry 将插桩、上下文传播和收集标准化，但云后端在存储、查询、访问、保留、告警和计费方面仍有差异。

Azure Monitor 将可观测性公开为可治理的资源：工作区、Application Insights、收集规则、诊断设置、告警、操作组、处理规则、仪表板和韧性测试。

SLI 衡量用户结果；SLO 定义目标；错误预算显式说明可容忍的不可靠性；消耗率把消耗转化为紧迫度。

告警应当路由一项可问责的决策，而不仅是宣布跨越阈值。

AIOps 可以划分优先级、进行关联、预测和辅助调查，但证据、作用域、权限和验证必须保持显式。

持续验证在暴露受控期间，把发布和配置变更连接到 SLO 证据。

混沌工程把可观测性作为有界、可证伪韧性实验的判定依据。

采样、基数、保留、隐私、工作区边界和摄取限制，使可观测性同时成为治理与 FinOps 议题。

多云可移植性依靠保留可观测性契约和证据模型实现，而不是假定产品等价。

参考资料与官方文档

来源说明：来源已于 2026 年 8 月 24 日核验。产品名称、工作区角色、预览或正式发布状态、支持区域、保留、摄取计费、AIOps 功能名称、混沌界面和 CLI 行为，都必须在书稿定稿时重新核查。

[1] OpenTelemetry. 可观测性入门。可观测性定义与未知之未知调查。

[2] OpenTelemetry. 什么是 OpenTelemetry？插桩、遥测信号与后端导出。

[3] OpenTelemetry. 信号。日志、指标、追踪及其他信号类别。

[4] OpenTelemetry. 上下文传播。跨进程与网络边界关联遥测。

[5] OpenTelemetry. 追踪。跨度、追踪身份与上下文传播。

[6] OpenTelemetry. Collector。供应商中立的接收、处理和导出管道。

[7] OpenTelemetry. 采样。头部与尾部采样概念及权衡。

[8] OpenTelemetry 规范。概述与客户端架构。

[9] Google.《站点可靠性工作手册》第 2 章：实施 SLO。

[10] Google.《站点可靠性工作手册》第 5 章：基于 SLO 告警。

[11] Google.《站点可靠性工作手册》附录 A：SLO 文档示例。

[12] Google.《站点可靠性工作手册》附录 B：错误预算策略示例。

[13] Microsoft Learn. Azure Monitor 文档与服务概述。

[14] Microsoft Learn. Azure Monitor 企业监控架构。

[15] Microsoft Learn. Azure Monitor 数据源与数据收集方法。

[16] Microsoft Learn. Azure Monitor 工作区概述，以及与 Log Analytics 工作区的区别。

[17] Microsoft Learn. Azure Monitor 中的指标与托管 Prometheus 数据平台。

[18] Microsoft Learn. 面向 .NET 应用的 Application Insights 与 Azure Monitor OpenTelemetry 指南。

[19] Microsoft Learn. Azure Monitor 告警概述与类型。

[20] Microsoft Learn. 创建 Azure Monitor 日志搜索告警规则。

[21] Microsoft Learn. 操作组、通用告警架构和告警处理规则。

[22] Microsoft Learn. Azure Monitor 告警最佳实践。

[23] Microsoft Learn. 保护 Azure Monitor 部署并使用 Azure Private Link。

[24] Microsoft Learn. Azure Chaos Studio 概述、Workspaces、Scenarios、报告、权限与当前预览限制。

[25] Azure CLI 参考。az monitor scheduled-query，版本 2.54.0 或更高。

[26] 阿里云文档。ARMS 用户指南与可观测数据存储概述。

[27] 阿里云文档。Managed Service for Prometheus 产品概述与 OpenTelemetry 指标摄取。

[28] 阿里云文档。ARMS 计费与摄取量限制。

[29] 腾讯云文档。腾讯云可观测平台概述与应用性能监控功能。

[30] 腾讯云文档。日志服务与腾讯云 Prometheus 监控服务。

[31] 腾讯云文档。APM OpenTelemetry 集成与链路协议标准。

[32] 腾讯云文档。混沌演练概述与实验工作流。

[33] 火山引擎文档。APMPlus OpenTelemetry 集成、日志服务、Kubernetes 可观测性与混沌工程实践。

第四篇 / 第 14 章

作为云控制平面的 FinOps

从成本可见性与分配，到护栏、单位经济性和经验证的节省

当成本与用量证据被规范化、分配、连接到业务价值、通过受治理的决策路由，并在技术或商业操作后得到验证时，FinOps 才成为一种工程能力。账单描述收取了哪些费用；FinOps 控制平面则解释谁拥有这项需求、它产生了什么结果、哪种干预是安全的，以及干预是否创造了持久价值。

本章内容

- 为何 FinOps 是托管控制回路，而不是成本削减运动或账单仪表板项目
- FinOps 框架如何连接理解用量与成本、量化业务价值、优化用量与费率，以及管理有效实践
- 为何云成本数据是延迟的、派生的、具有合同效力的证据，而非实时计数器
- 账单作用域、资源作用域、FinOps 作用域、实际成本、账单成本、有效成本和摊销成本如何回答不同问题
- FOCUS 如何跨供应商和技术类别提供规范化的成本与用量契约
- 分配、标签、成本中心、共享成本规则、成本展示和成本分摊如何建立可问责的所有权
- 为何预算、预测、异常、配额、额度、策略、预留和节省计划是不同的控制机制
- 用量优化、费率优化、架构、工作负载放置和需求设计如何构成优化组合
- 单位经济性如何将资源消耗连接到业务结果，包括 AI 原生和智能体式工作负载
- Azure 成本管理、Advisor、预算、导出、分配、配额和承诺工具如何构成参考 FinOps 控制平面
- Azure、阿里云、腾讯云和火山引擎如何通过不同的账单与资源模型表达相同的 FinOps 意图
- 如何创建资源组预算、查询成本证据、组装可审计证据包，并在无需等待真实超支事件的情况下完成清理

云账单即使完全准确，在运行上也可能毫无用处。它可以告诉组织本月支出增加，却无法指出是哪项产品功能、租户、环境、部署、模型或工程决策导致了变化。它可以显示折扣，却不能证明承诺得到了充分利用。它可以显示未达到服务目标的低成本架构，或产生

了成比例更高业务价值的高成本架构。财务记录不可或缺，但它还不是一套工程控制系统。

FinOps 提供了缺失的运行模型。FinOps Foundation 将 FinOps 定义为一种运行框架和文化实践：它通过工程、财务和业务团队协作，使技术业务价值最大化，支持及时的数据驱动决策，并建立财务问责 [1]。本章把“FinOps 控制平面”作为一种编辑性的工程视角，而非对 Foundation 定义的替代。它指连接成本与用量证据、所有权、策略、操作、验证和学习的托管层。

期望的回路不是“看到一个大数字，然后削减某些东西”，而是“理解发生了什么变化，将其分配给可问责的作用域，量化由此产生的价值，选择安全的技术或商业响应，并依据成本、需求、可靠性、安全和产品结果验证响应”。云使之成为可能，因为资源、计费计量项、身份、策略、活动日志、预算、配额、导出和建议本身都是可编程的控制平面对象。

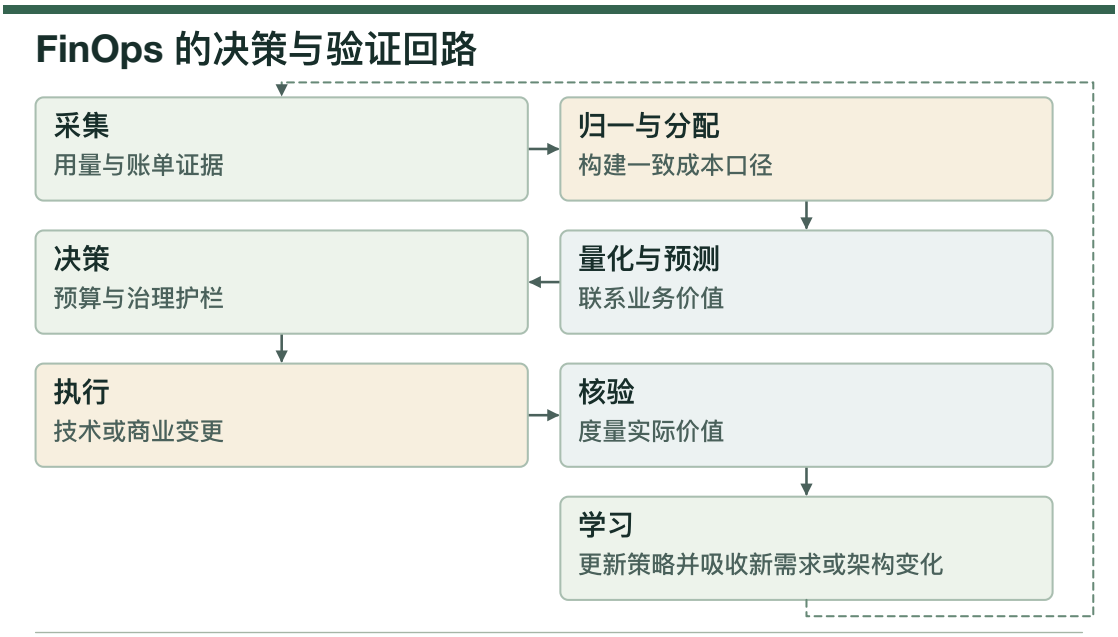


图 14-1 FinOps 作为托管反馈回路。只有当成本与用量证据得到分配、连接到价值、通过受治理的决策进行路由、付诸行动并得到验证时，它才真正有用。

14.1 FinOps 是控制平面，而非成本削减运动

降低成本可以是 FinOps 的一种结果，却不是定义。团队可能通过移除冗余、抑制可观测性、改用较弱模型或推迟安全工作来降低月度账单。数字下降了，业务风险却在上升。相反，增长中的服务可能因为服务更多客户、改善延迟、提高检索质量或替代昂贵的人工工

作而增加支出。数字上升了，单位经济性却在改善。因此，FinOps 关注技术价值和决策质量，而不是最小化某个总额。

控制平面视角阐明了必须管理的内容。控制平面并不执行每项工作负载操作，而是定义资源、身份、作用域、期望状态、策略与证据。同样，FinOps 控制平面不应集中做出每项工程或采购决策。它提供共同的成本与价值模型，在消费点暴露决策信号，并为分布式团队建立安全的行动路径。

表 14-1 相关财务与工程责任

概念	首要目的	孤立使用时的典型失效方式
计费与开票	生成具有合同效力的费用、额度、税款和付款记录。	财务上准确的记录到达太晚，或缺少工程行动所需的工作负载上下文。
成本管理	分析、分组、预测、告警、分配和导出成本与用量数据。	仪表板变成了没有可问责所有者或变更机制的报告。
优化工具	识别潜在的用量、费率或架构改进。	建议被盲目接受、无人行动，或其收益从未得到验证。
FinOps 实践	协调工程、财务、产品、采购和领导层对技术价值的认识。	成本成为专家关注的问题，而非架构与产品约束。
FinOps 控制平面	把证据、所有权、护栏、操作和验证连接为可编程的运行策略。	自动化在缺少上下文的情况下行动，或组织无法解释为何作出某项成本决策。

14.1.1 集中赋能，分布式所有权

FinOps 应集中赋能，并在技术决策发生的地方实践。中央团队可以定义架构、分配规则、承诺策略、数据质量、通用 KPI 和获批自动化，却无法知道每个工作负载的延迟目标、季节性、客户合同、模型质量阈值或风险容忍度。产品和工程团队必须负责连接自身的需求、架构、成本和结果。

这种运行模型类似平台工程。平台团队提供铺好的道路：规范化数据、自助式视图、预算模板、策略模块、建议契约和可审计的变更工作流。工作负载团队使用这条道路，同时仍对自身决策的正确性负责。仅发送月度电子表格无法实现财务问责；只有当成本信号出现在设计评审、积压工作优先级、发布门禁和运行例程中时，问责才真正建立。

注意：FinOps Foundation 资料采用 CC BY 4.0 许可。本章将 Foundation 框架改编为工程控制平面模型，并在参考资料中注明来源。

14.2 运行模型：理解、量化、优化和管理

当前 FinOps 框架描述四项业务结果：理解用量与成本、量化业务价值、优化用量与成本，以及管理 FinOps 实践 [2][3]。熟悉的“告知、优化、运行”阶段作为迭代运动仍然有用，但这些领域更清楚地说明预期结果。组织必须理解证据，将证据连接到价值，改善技术与商业决策组合，并维护使实践可靠的机制。

表 14-2 以控制平面结果表达的 FinOps 领域

领域	控制平面问题	代表性托管能力
理解用量与成本	消费了什么、收取了什么费用、如何分配，以及发生了什么变化？	数据摄取、规范化、分配、报告、分析、异常管理、数据质量。
量化业务价值	支出促成了什么结果？它是否符合预期？	规划、估算、预测、预算、KPI、基准比较、单位经济性。
优化用量与成本	哪个用量、费率、架构或放置决策能够改善价值？	合理调整规模、弹性、计划、生命周期策略、承诺、许可、工作负载放置。
管理 FinOps 实践	谁拥有决策、适用哪些策略，以及如何保存证据？	治理、教育、自动化、工具、采购协调、成熟度和运行节奏。

这些领域不是瀑布流程。一次优化尝试暴露出所有权缺失后，分配质量可能随之提高；预算评审可能发现某个产品指标不适合作分母；承诺购买可能暴露需求预测错误。设计良好的控制平面会保留这些反馈路径，而不是假装成本管理沿着一条不可逆序列前进。

14.2.1 成本是设计输入，而非部署后报告

杠杆最大的 FinOps 决策往往发生在资源存在之前。区域选择会影响价格、延迟、出站流量、可用性和数据驻留。托管服务的单位成本可能更高，却减少人工和事件风险。Serverless 设计可以减少闲置容量，却增加单次请求成本或启动延迟。更大的模型可能把质量提高到足以减少重试和人工审查。正确决策需要综合考虑成本、速度、质量、可靠性、安全和战略约束。

这就是 FinOps 属于架构评审、服务级别设计和产品规划的原因。云的可变成本模型使架构在经济上可观测：设计可以在发布后测量和调整。但只有当组织在部署前定义证据契约时，这项收益才会出现——所有权标签、成本作用域、单位指标、预算预期、质量目标，以及验证变更所需的信号。

14.3 成本数据是延迟、派生且具有合同效力的数据集

运行遥测与账单证据从不同视角描述同一系统。请求追踪可能在数秒内可用，相应计费量却可能稍后才按费率计算、根据合同调整、跨时间摊销、转换为账单货币，并按日或按月导出。实时仪表板可以估算成本，但发票和供应商计量记录仍是财务事实来源。FinOps 工程必须同时建模速度与权威性。

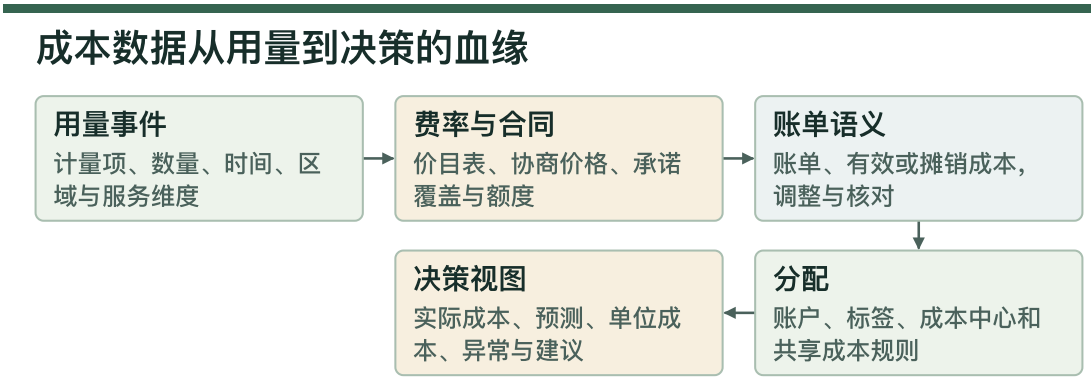


图 14-2 成本数据沿袭关系。用量经过定价与合同语义、计费规则、分配策略和报告视图的转换，最终成为决策证据。

14.3.1 作用域是每项成本事实的一部分

没有作用域的数字含义不明。Azure 公开账单帐户、账单配置文件、发票科目和注册帐户等账单作用域，也公开管理组、订阅和资源组等资源管理作用域。FinOps 作用域则可能代表产品、业务单元、租户类别、环境、AI 计划或成本中心。这些结构会重叠，但含义并不相同。

每项查询、预算、预测、建议和单位指标都应记录其作用域标识符与类型。如果共享平台服务计费到其他地方，就不能假定资源组预算代表产品；如果多个租户共享推理、搜索、存储或可观测性资源，也不能把订阅总额当作租户成本。因此，作用域设计是一项建模决策，而不仅仅是筛选器选择。

14.3.2 成本语义回答不同问题

表 14-3 必须显式选择成本语义

成本视图	它回答的问题	常见误用
账单或发票成本	账单周期内具有合同效力的收费金额是多少?	把它用作实时信号，或把所有共享费用直接归因于工作负载。

成本视图	它回答的问题	常见误用
实际或用量成本	报告期间有哪些费用与用量相关?	忽略后续调整、退款、额度或承诺处理。
摊销或有效成本	应在覆盖用量和时间之间分配的经济成本是多少?	把摊销当作现金流，或期望它与发票行完全一致。
标价或按需成本	按参考标价计算时，用量成本是多少?	把标价与协商价格的差额表述为有保证的节省。
预测成本	如果当前行为和假设持续，可能花费多少?	把预测当作承诺，或不记录模型置信度和季节性。
单位成本	定义的资源或业务结果成本是多少?	随时间改变分母或成本边界，却不为指标设置版本。

除非语义明确，否则设计文档和 API 中应避免使用“成本”这一笼统说法。成本查询契约应标明货币、成本类型、定价期间、使用间隔、时区、税务处理、额度、承诺摊销和分配版本。当 Agent 汇总支出或建议操作时，这种精度尤其重要：一个貌似简单的总额可能混合了不应比较的值。

14.3.3 数据新鲜度与核对是工程要求

成本数据并不保证以遥测速度到达。Azure 文档指出，成本和用量数据通常在 8 至 24 小时内可用，预算每 24 小时评估一次 [8]。阿里云及其他平台同样记录了某些成本视图按日或按结算周期产生延迟。控制平面必须公开新鲜度、完整性与结算状态，而不能悄无声息地把最新可用数字当作当前数字返回。

需要亚小时级响应的运行控制应使用服务遥测、配额、速率限制和工作负载特定计数器，随后再由成本数据核对财务影响。例如，AI 网关可以实时施加每租户令牌速率限制，第二天再由供应商账单确认计费计量项。两条证据路径应彼此关联，而不应混淆。

14.3.4 FOCUS 作为规范化的成本与用量契约

FinOps 开放成本与用量规范（FOCUS）定义与 FinOps 术语一致的通用账单数据架构。在本章来源核验日期，1.4 版是当前已发布规范。它支持成本分摊、分配、预算、预测、承诺分析和发票核对等通用能力 [4][5]。

FOCUS 不会抹去供应商差异，而是提供一个稳定语义边界，用于规范化供应商特定数据。组织仍需记录符合程度、不支持的列、供应商扩展、导出频率、货币行为和核对规则。因此，FOCUS 数据集是可移植分析契约，而不是每个供应商都以完全相同方式表示每个概念的证明。

表 14-4 可移植成本契约的最小证据属性

必需数据属性	重要原因
供应商与帐户身份	防止相同资源名称或 SKU 在不同云和账单上下文中发生冲突。
资源与服务身份	将计费用量连接到清单、所有权、配置和生命周期证据。
收费期间与使用间隔	区分用量发生时间与供应商收费或调整时间。
账单成本与有效成本	支持发票核对和具有经济意义的分配。
定价与承诺上下文	解释收费费率为何不同于标价或按需价格。
标签与分配元数据	把供应商记录连接到产品、团队、环境、租户和成本中心。
货币与汇率处理	防止跨账单货币或转换日期进行无效聚合。
新鲜度与完整性状态	使自动化能够判断数据是否足以用于调查或财务结账。

14.4 分配：将共享支出转化为可问责所有权

没有分配的成本可见性只会产生一个共享总额和一个共同借口。分配将成本与用量指派给可问责的业务构造：产品、平台、团队、成本中心、租户、环境或计划。它是供应商账单与分布式工程所有权之间的桥梁。

没有任何单一分配机制足够。帐户层次提供强结构边界，却可能造成管理碎片。资源组和项目与部署生命周期一致，但可能不符合产品所有权。标签灵活但不完整且可变。成本中心和分配单元可以重新分配共享费用，却依赖文档化规则。成熟的分配会组合这些机制，并为把供应商记录转换为管理视图的规则设置版本。

14.4.1 从确定性所有权开始，再细化共享成本

1. 通过帐户、订阅、项目、资源组、资源 ID 或供应商支持的项目归因，为拥有直接且稳定所有者的成本指派归属。
2. 对层次结构无法表示的业务元数据使用必需标签，例如产品、环境、所有者、租户类别或实验。
3. 显式识别共享平台服务，而不是将其隐藏在未分配余量中。
4. 依据可观测驱动因素——请求、存储、活跃用户、CPU 时间、令牌或其他因果度量——选择共享成本规则，而非武断地平均分摊。
5. 保留分配规则、版本、生效日期、源字段和未分配金额，使结果可以审计和复现。

未分配存储桶不仅是报告上的不便，也是一项数据质量指标。未分配比例增长意味着所有权减弱、新服务没有进入资源分类法，或标签在成本发生后才创建。控制平面应衡量这种情况，并把它路由给负责纠正的平台与财务所有者。

14.4.2 成本展示与成本分摊是治理选择

成本展示向所有者报告成本，但不更改内部财务帐户。成本分摊则把成本转移或记入责任单元。二者依赖相同的分配证据，却会产生不同的激励和核对要求。团队可以先采用成本展示来提高数据质量和建立信任，之后再采用成本分摊。如果在分配逻辑仍有争议时过早分摊成本，技术数据问题就会变成组织冲突。

表 14-5 分配机制及其失效方式

分配层	强项	主要风险
帐户/订阅/项目	硬管理与账单边界；委托所有权。	数量激增、重复的共享服务和碎片化承诺。
资源组/命名空间	生命周期、环境或应用分组。	资源可能放错位置；跨组依赖仍然共享。
标签	灵活的业务元数据与多云转换。	缺失、不一致、追溯应用或由用户控制的值。
成本中心/分配单元	管理层次与共享成本再分配。	规则不透明、重复分配或因果依据薄弱。
用量驱动因素	按请求、席位、存储、令牌或计算时间进行因果分摊。	插桩缺口，以及关于正确分母的争议。
手动调整	例外纠正与财务结账。	丧失可复现性并越来越依赖电子表格。

14.5 护栏：预算、预测、异常、配额、额度、策略与承诺

云控制台经常把多种控制放在相近位置，容易让人错误地认为它们等价。预算依据计划跟踪支出；预测估算未来可能的支出；异常检测反常行为；配额限制或分配服务容量；额度或支出限制代表商业余额或帐户约束；策略控制允许的资源配置；预留或节省计划改变合格用量所适用的费率。每种工具处理不同的失效方式。



图 14-3 预算、配额、额度、策略、承诺和异常信号是互补护栏，而非可以互换的限制。

14.5.1 预算是问责路径，而非终止开关

Azure 预算可以针对实际或预测阈值发出告警，但预算不会停止资源 [8]。成本数据延迟也意味着在评估观察到阈值之前，工作负载可能继续产生支出。这种行为对许多生产系统是正确的：在财务阈值处突然关闭数据库或 API，可能造成比超额成本更大的业务损失。

有用的预算应当定义可问责路径。在早期阈值处，所有者收到上下文和预测；在更高阈值处，必须评审；在关键阈值处，获批的操作组或工作流可以限制非生产需求、暂停实验、要求新资源获得显式批准，或调用工作负载特定的安全模式。响应体现的是架构和策略，而不是假定账单对象会强制执行运行时安全。

14.5.2 配额是容量治理

配额是订阅、区域、模型或资源类型内的服务额度或限制。Azure 指出，申请增加配额不产生费用；费用来自资源使用，而非配额本身 [15]。配额既可以防止失控扩展，也可能阻止合法部署。配额监控属于 FinOps，因为容量与支出相互作用，但配额不是预算。

AI PaaS 使这一区别尤其重要。Microsoft Foundry 配额与限制在订阅、区域、模型或部署类型作用域定义，而不是作为租户级财务强制机制 [18]。即使底层模型平台提供供应商级配额，服务多个租户的应用仍必须实施自己的归因、速率策略和租户护栏。

14.5.3 承诺优化费率，而非架构

预留和节省计划可以降低可预测合格用量的有效费率。Azure 明确把预留描述为一种不影响资源运行时状态的账单折扣 [16]。预留不会创建虚拟机、提高利用率或纠正规模过大的架构；它可能只是在锁定财务承诺的同时让浪费变得便宜。

承诺决策应以需求预测和风险策略为依据。覆盖率衡量有多少合格用量获得折扣；利用率衡量已购买承诺有多少被实际消费；灵活性衡量工作负载是否可能改变区域、服务、系列或运行模型。采购、财务和工程必须共同评估这些维度。

14.5.4 异常检测是调查信号

异常系统将当前行为与预期模式比较并呈现反常变化。Azure 成本管理每日依据规范化用量识别异常，而供应商文档通常会警告异常结果可能延迟、不完整或仅供参考 [9][26]。成本异常可能源于缺陷、攻击、缺失的生命周期策略、新客户、成功的营销活动、价格变化或账单调整。

控制平面的响应是用所有权、部署变更、资源清单、利用率、业务需求和服务级别证据丰富异常。异常应以假设与证据发起调查，而不应自动删除看起来昂贵的资源。

14.6 优化是技术与商业决策组合

FinOps 优化应组织为组合，而不是折扣技巧清单。FinOps 框架区分用量优化、费率优化、架构与工作负载放置、许可及相关能力。阿里云架构框架的成本框架同样区分计费方式优化、资源使用优化和架构优化 [21]。这些杠杆相互影响，应依据工作负载价值与风险进行评估。

表 14-6 优化不只是合理调整规模

优化杠杆	示例	所需验证
需求与产品设计	减少不必要调用、重复数据、未使用功能、过度重试或低价值的模型扇出。	结果数量和质量仍可接受；需求假设保持显式。
用量效率	合理调整规模、计划关机、自动伸缩、释放闲置资源、应用存储生命周期、调优保留期。	在不违反延迟、可用性、恢复或安全目标的前提下提高利用率。
架构与放置	选择托管或自管理、区域、Serverless 或预配、批处理或在线、缓存、队列、模型路由。	在具有代表性的工作负载期间，总拥有成本和运行风险得到改善。
费率优化	预留、节省计划、承诺使用、资源包、协商费率。	覆盖率和利用率满足策略；锁定与预测风险已得到理解。
运行纪律	检测孤立资源、使实验到期、强制所有权、自动清理与评审。	控制可复现、可审计，并且不会移除工作负载仍需要的资源。
供给与平台设计	提供获批 SKU、黄金路径、默认预算、成本感知 API 和可复用策略。	团队可以选择成本有效的默认值，而无需绕过可靠性或合规。

14.6.1 建议是带有变更契约的假设

建议应包含受影响作用域与资源 ID、证据窗口、当前配置、利用率或需求证据、预期节省语义、置信度、风险、依赖、所需审批、拟议变更、回滚条件和验证计划。“通过调整规模节省 500 美元”不是工程建议，而是缺少安全论证的估算。

建议还必须说明排除哪些因素。供应商估算可能考虑计算计量节省，却没有考虑迁移人工、软件许可、预留破损、数据传输、运行劳务、质量损失或事件暴露。FinOps 工程让这些边界可见，使人类或策略引擎能够诚实比较替代方案。

14.6.2 优化系统，而非单个行项

局部成本优化可能增加系统成本。压缩日志会降低摄取，却增加 CPU 并延迟事件诊断。把数据迁移到更便宜的区域可能增加出站费用和延迟。选择最小模型可能增加重试、工具调用、人工升级或失败任务。移除冗余会降低计算支出，却可能耗尽错误预算。

因此，安全的控制平面会连接成本证据与可观测性和可靠性。每项重大优化都应指出必须保持在边界内的 SLO 与业务指标。这使 FinOps 成为架构和运维的延伸，而不是独立的月度评审。

14.7 单位经济性：每项结果成本，而非每项资源成本

资源总额解释资金被计费在何处；单位经济性则解释系统能否高效地把技术转化为价值。FinOps Foundation 区分资源效率指标（如每 GB 或每令牌成本）与业务单位指标（如每租户、交易、案例或结果成本）[4]。如果成功结果增长更快，总成本上升可能是健康的；如果质量或采用率崩溃，总成本下降也可能有害。

只有分子和分母都经过定义并设置版本时，单位指标才有效。分子必须指明成本边界：仅直接云费用、摊销承诺、共享平台分配、支持、可观测性、工程人工或其他完全加载成本。分母必须代表有意义且经过验证的单位，而不是容易操纵的活动计数。

```
unit_cost(scope, period, version) =
    allocated_effective_cost(scope, period, allocation_version)
    / verified_outcome_units(scope, period, outcome_definition_version)
```

14.7.1 资源单位是垫脚石

每 vCPU 小时成本、每 GB 存储成本、每百万令牌成本、每次搜索查询成本或每次函数调用成本等资源指标很有用，因为工程师可以直接影响它们。它们揭示费率变化、利用率和服务效率，却不能独自证明业务价值。如果应用为完成同一任务使用三倍令牌，更便宜的令牌也没有价值。

14.7.2 AI 原生系统需要完全加载的边界

AI 应用成本经常看似只有模型 API 账单。实际上，一个用户请求可能触发提示处理、检索、嵌入、重排、内容安全、多次模型调用、工具执行、排队、存储、网络传输、可观测性、评估和人工审查。Microsoft Foundry 文档明确指出，Foundry 成本只是整个应用或解决方案成本的一部分 [17]。



图 14-5 完全加载的 AI 单位经济性。模型推理只是其中一层；分母应当是经过验证的业务结果。

表 14-7 AI 单位指标应向经过验证的结果演进

候选 AI 单位	适用情形	失效方式
每输入/输出令牌成本	比较计量项、路由、缓存或提示行为。	只优化令牌价格，却忽略任务成功、重试或下游工作。
每次模型请求成本	调用具有可比的有效负载与质量要求。	一次请求的令牌数或推理工作量可能截然不同。
每个有依据答案的成本	明确 RAG 质量与引用要求。	把缺少支持、错误或被用户拒绝的答案也计入。
每项获批智能体操作的成本	工具执行可审计，并产生经过验证的操作。	忽略被拒绝、回滚、重复或不安全的操作。
每个已解决案例的成本	AI 替代或辅助可衡量的工作流结果。	把全部解决价值归因于 AI，却不考虑人工工作或案例复杂度。

候选 AI 单位	适用情形	失效方式
每租户/活跃用户成本	共享 SaaS 成本需要成本展示或定价分析。	平均值掩盖重度用户和跨租户补贴。

14.7.3 单位经济性与 SLO 应一同评审

有用的运行视图会把单位成本与吞吐量、质量、延迟、可用性和错误预算消耗结合。路由变更可能降低每个答案的成本，却增加 P95 延迟。更大的模型可能提高直接成本，却提高首次成功率并降低每个已解决案例的成本。检索缓存可能降低模型与搜索成本，却增加陈旧风险。显式权衡可以防止 FinOps 优化掉产品本身。

14.8 Azure 作为参考 FinOps 控制平面

Azure 被选作参考平台，是因为它把账单层次、资源层次、成本分析、分配、预算、异常信号、导出、建议、承诺工具、配额、策略、身份和活动证据公开为彼此连接的控制平面能力。具体门户标签和 API 版本可能变化；架构则更加稳定。



图 14-4 Azure 参考 FinOps 控制平面。成本管理把资源与账单证据转换为决策视图，治理和工程工作流则采取行动并验证。

14.8.1 账单作用域、资源作用域与 FinOps 作用域

Azure 成本 API 在受支持的作用域运行。账单帐户或账单配置文件支持财务核对；管理组或订阅支持企业治理；资源组支持工作负载或实验生命周期。成本管理查询操作直接接受这些作用域标识符。随后，FinOps 模型可以利用标签和分配逻辑，把记录重新分组为产品、租户、成本中心或计划作用域。

必须在任务所需作用域授予权限。成本读取者不需要调整资源规模或删除资源的权限；预算作者并不自动需要应用数据访问权；建议执行者应使用独立身份、显式审批策略和可审计变更路径。FinOps 中的最小权限就是分离观测、决策与操作。

14.8.2 成本分析、查询与导出

成本分析提供实际成本和摊销成本的交互式视图，并可按资源、服务、标签及其他维度查看。程序化查询可以把定义的作用域、时间范围、聚合、分组和筛选器编码进 API 请求。计划导出把成本数据集写入存储，以用于更大规模分析和核对。Azure 架构良好的框架建议每日导出到 ADLS Gen2，并使用 FOCUS 模板进行供应商中立分析 [10]。

交互式视图回答调查问题；导出支持可重复管道与历史分析。控制平面设计应保存查询或导出定义以及产生的证据。缺少作用域、时间窗口、语义或筛选器的屏幕截图无法复现。

14.8.3 分配、标签继承与共享成本

Azure 成本管理可以应用标签继承并拆分共享成本 [9]。标签继承填充成本报告元数据，不应与修改底层资源标签混淆。分配规则是对成本数据的管理转换，而不是对已部署资源图的变更。因此，该规则必须设置版本，并与基础设施分开核对。

标签应在接近资源创建的位置通过策略与 IaC 治理。成本分配可以弥补缺口，但不应成为业务身份的永久来源。最可靠的设计会结合稳定层次、必需标签和显式共享成本规则。

14.8.4 预算、预测、异常与告警

预算是具有生命周期、作用域、期间、金额和通知定义的资源。它可以评估实际或预测阈值。成本告警提供预算、额度、支出配额及其他告警类别的统一视图。异常检测则提供独立的每日调查信号。FinOps 控制平面应把这些信号路由给所有者，并在适当时路由给获批工作流，而不是发送没有区分的电子邮件。

14.8.5 Advisor、承诺与配额

Azure Advisor 包含面向订阅与资源组的成本建议。应把这些建议视为供应商生成的假设：行动前验证工作负载行为、依赖、SLO、承诺状态和业务需求。预留和节省计划处理费率；配额处理容量；两者都不能替代用量优化或架构评审。

这一区分对运维很重要。购买预留可以降低费率，却不会改变规模过大的资源；提高配额可以支持增长，而不立即产生费用；降低内部租户限制可以控制应用需求，即使供应商配额仍然很高。每项决策属于不同的控制界面。

14.8.6 FinOps 证据链

表 14-8 FinOps 对象应产生可审计证据

控制平面对象	应保存的证据
成本查询或已保存视图	作用域、时间范围、成本语义、货币、分组、筛选器、API 版本、数据时间戳。
导出	定义、架构/FOCUS 版本、计划、存储目的地、完成状态、核对状态。
预算	作用域、金额、期间、阈值类型、收件人或操作组、部署身份、创建事件。
分配规则	源作用域、目标作用域、驱动因素、公式、版本、生效日期、未分配余量。
建议	资源 ID、证据窗口、当前状态、预期节省、风险、假设、所有者、到期时间。
承诺	购买作用域、期限、合格服务、覆盖率、利用率、交换/退款策略、所有者。
优化变更	获批计划、变更身份、部署记录、回滚条件、变更后证据。

14.9 AI PaaS 与 Agent 成本控制

AI PaaS 引入新计量项和新的放大路径。令牌消耗可变，但令牌价格只是一个组成部分。推理模型可能在响应前生成隐藏的推理令牌。一个用户问题可能分支为多次检索查询、模型调用、安全检查和工具执行。评估管道可能重放数千个样例。Agent 可能重试失败工具或陷入循环。知识刷新可能重新嵌入整个语料库。FinOps 必须建模从一次业务请求到完整资源图的扩展过程。

14.9.1 计量语义与部署经济性

Foundry 模型可能采用按令牌付费、预配吞吐量、托管计算或供应商特定容量模型。计量项和部署类型决定成本行为。HTTP 状态本身未必决定是否计费；Microsoft 建议将服务指标与成本管理计量数据核对，并把发票和计量记录视为权威 [17]。

成本感知的 AI 网关应记录模型、部署、提示与补全令牌、公开的缓存或推理行为、延迟、重试次数、租户或项目、请求目的、质量结果以及调用的下游资源。这些应用证据支持快速控制与分配；供应商账单随后核对收费成本。

14.9.2 供应商配额不是租户隔离或租户预算

Microsoft Foundry 文档指出，配额与限制不在租户级强制执行；最高级别限制与 Azure 订阅和模型相关作用域关联 [18]。因此，多租户应用必须实现自己的租户归因、速率限制、预算、审批规则和跨租户隔离。供应商配额保护服务容量；应用策略保护客户与产品边界。

14.9.3 AI 成本扇出

表 14-9 AI 原生成本放大路径

扇出来源	成本风险	控制平面响应
长上下文与无界历史	令牌增长随每轮对话复合累积。	汇总、选择性检索、限制上下文、为记忆策略设置版本。
Agent 规划与重试	一个用户请求创建许多模型和工具调用。	最大步骤数、重试预算、超时、循环检测、每任务成本包络。
检索扩展	多查询、重排、嵌入和较大 top-k 会增加成本。	查询计划限制、缓存、检索质量评估、每个有依据答案的成本。
评估与红队测试	大型数据集重复重放模型调用。	评估预算、采样、计划窗口、结果复用、完整运行审批。
工具执行	函数、容器、SaaS API、出站流量和人工审查会增加成本。	工具作用域、价格元数据、试运行、审批、结果与收费证据。
可观测性	高基数提示、追踪和有效负载增加摄取与隐私风险。	采样、编校、保留层级、成本归因、证据最低要求。

14.9.4 AI 工作的成本包络

成本包络是附加到任务、租户、项目、评估运行或 Agent 实例的应用级策略。它可以包含最大令牌数、模型类别、检索查询、工具调用、已用时间、重试、估算货币金额和超过阈值所需的审批。包络利用应用遥测和供应商配额实时执行，随后再用账单证据核对。

```
ai_cost_envelope = {
  "tenantId": "tenant-a",
  "taskType": "finops-investigation",
  "maxModelCalls": 6,
  "maxToolCalls": 8,
  "maxInputTokens": 60000,
  "maxOutputTokens": 12000,
  "allowedModels": ["economy", "standard"],
  "estimatedCostLimit": 3.00,
```

```
  "approvalRequiredAbove": 1.50,
  "qualityGate": "grounded-and-cited"
}
```

该包络不是 Azure 预算，而是一项能够立即响应的应用控制。随后的 FinOps 管道把供应商费用分配给同一租户和任务，比较估算成本与账单成本，并更新路由与估算策略。

14.10 从潜在节省到经过验证、实现的价值

供应商建议和内部分析通常报告潜在节省。潜在节省是有条件的估算：它依赖需求、合格小时数、费率、作用域，以及变更能够安全执行的假设。实现的节省是在获批变更后观测到、针对需求与服务结果进行规范化，并与成本证据核对的节省。



图 14-6 建议生命周期。只有在获批变更得到观测，并依据成本、需求和服务证据验证后，节省才真正实现。

14.10.1 建立经得住审查的基线

基线应覆盖代表性期间，并记录需求、季节性、部署版本、费率、承诺覆盖率、区域、SLO 和已知事件。比较节假日周与普通周，或把失败服务与健康服务比较，可能制造虚假节省。当需求发生变化时，应使用单位指标或规范化模型，而非原始总额。

```
realized_saving =
    expected_cost_for_post_change_demand_under_baseline_policy
    - observed_effective_cost_after_change

仅在以下情况下有效：
    SLOs、quality gates、security controls 和 scope definitions 仍得到满足
```

14.10.2 保存变更与验证证据

表 14-10 声称实现节省所需的证据

阶段	所需证据
检测	成本查询、利用率、所有者、作用域、数据新鲜度、相关部署与需求变更。
提案	预期节省语义、模型、假设、风险、排除成本、到期时间、回滚计划。
审批	审批者、策略、职责分离、维护窗口、操作哈希或变更 ID。
执行	IaC/GitOps 部署、身份、活动日志、旧配置与新配置。
观测	成本、单位指标、SLO、质量、事件、用户需求和承诺覆盖率。
验证	实现的节省、与估算的偏差、副作用、持续期间、经验与策略更新。

这种证据模型使优化成为可审计的工程实践，也支持自动化而无需授予 Agent 无界权限。只读分析和建议可以广泛委派；变更操作则需要独立身份、策略门禁、审批、有界爆炸半径和回滚。

14.11 多云转换：保留契约，而非控制台

主要云平台通过不同的帐户层次、成本中心、账单数据集、预算系统、承诺产品和 API 来公开相似的 FinOps 意图。可移植性来自保留证据与决策契约——作用域、成本语义、分配、所有者、护栏、操作和验证——而非假定产品名称或控制台页面等价。

表 14-11 多云 FinOps 能力映射

能力	Azure	阿里云	腾讯云	火山引擎
成本分析	成本管理/成本分析	费用与成本中的成本分析	成本分析	账单中心的成本分析
分配	层次、标签、标签继承、分配规则	成本中心、层次、标签、共享成本分配	成本分配标签与成本分配单元	成本单元与分配规则
预算	带实际/预测通知的成本预算	带告警的成本、用量/覆盖率预算	预算管理	账单与预算能力因帐户/产品而异；验证当前支持情况
异常/监控	每日成本异常信号与成本告警	基于规则的监控与 AI 辅助异常检测	预算与未结算用量监控；验证当前异常功能	成本/用量视图和告警；验证当前异常功能

能力	Azure	阿里云	腾讯云	火山引擎
导出	计划导出，包括 FOCUS 模板	账单和成本明细导出/API	账单导出到 COS 和导出中心；FOCUS 1.0 及文档化缺口	账单数据存储到 TOS 和账单 API
承诺	预留和节省计划	节省计划、订阅、资源包	按服务提供的预留或订阅折扣	按服务提供的节省计划和订阅选项
配额	Azure Quotas 和服务特定配额告警	服务配额与配额管理	服务配额与用量限制	配额中心与配额告警
身份	Azure RBAC 与成本管理角色	RAM 与账单权限	CAM 与账单中心角色	IAM 与账单中心权限

14.11.1 阿里云

阿里云成本中心可以按层次组织成本并分配共享成本。预算管理支持实际与预测告警，也针对部分承诺场景公开用量或覆盖率导向的预算。成本分析和异常相关功能提供调查信号。阿里云架构框架明确把 FinOps 描述为循环往复的周期，并区分计费方式、资源用量和架构优化 [21]。

14.11.2 腾讯云

腾讯云提供成本分析、预算管理、成本分配标签、成本分配单元、分账单、导出机制，以及公开的 FinOps 成本管理解决方案。腾讯云还发布了 FOCUS 1.0 数据集并记录符合性缺口。这项披露很重要：规范化格式应附带供应商特定的符合性声明和核对测试，而不应假定其完整。

14.11.3 火山引擎

火山引擎账单中心文档包含成本单元、成本分析能力、账单数据存储到 TOS、账单 API、配额中心、配额告警，以及服务特定节省计划或订阅模型。产品可用性和英文文档覆盖程度不一，因此生产比较应记录帐户类型、区域、产品、计费模式和文档核验日期。

14.11.4 可移植 FinOps 契约

```
finops_record = {
  provider, billing_account, resource_scope, finops_scope,
  usage_interval, charge_period, currency, cost_semantic,
  billed_cost, effective_cost, allocation_rule_version, owner,
  business_unit_metric, data_freshness, source_evidence
}

finops_decision = {
```

```
  trigger, hypothesis, expected_value, risk, policy, approver,
  action, rollback, verification_window, realized_value_evidence
}
```

该契约可以用 FOCUS 和供应商扩展实现，但应在应用与数据模型中保持显式。它是使多云工具、仪表板和 Agent 能够推理可比意图，同时不抹去供应商语义的边界。

14.12 能力检查 C3：创建预算并捕获控制平面证据

这项独立能力检查创建一个 Azure 资源组预算作为托管控制平面对象。它不需要应用、之前的实验或真实超支事件。目标是证明生命周期、作用域、通知策略、程序化成本查询、证据收集和清理。

14.12.1 学习目标

- 创建带有所有权和成本中心标签的专用资源组。
- 使用 Bicep 在资源组作用域部署月度预算。
- 同时定义实际与预测通知。
- 通过 Azure Resource Manager API 获取预算资源和成本查询。
- 记录数据新鲜度限制，并说明预算为何不是运行时终止开关。
- 组装证据包，并显式删除预算与资源组。

14.12.2 先决条件与成本边界

- 一个 Azure 订阅，并且你可以在其中创建资源组和该作用域的成本管理预算。
- 已安装 Azure CLI 与 Bicep，且帐户已登录预期租户和订阅。
- 仅用于实验通知定义的一个电子邮件地址。
- 不需要可计费工作负载。预算本身不会产生用量成本。成本查询可能返回空行或延迟数据。

注意：不要为了强制触发告警而降低预算并故意创建昂贵资源。Azure 成本数据通常延迟 8 至 24 小时，预算每 24 小时评估一次。证据目标是控制平面配置和查询契约，而不是人为制造的账单。

14.12.3 定义实验变量

```
export sub=$(az account show --query id -o tsv)
export location=eastus
export rg=book-finops-c3
export budget=book-finops-monthly
export owner_email="you@example.com"

az group create \
```

```
  --name "$rg" \
  --location "$location" \
  --tags owner="$owner_email" costCenter=book-labs environment=learning
```

记录资源组 ID 与标签。资源组同时是本次能力检查的管理作用域和清理边界。

```
export scope="/subscriptions/$sub/resourceGroups/$rg"
az group show --name "$rg" --query "{id:id,tags:tags,location:location}" -o
jsonc
```

14.12.4 创建 Bicep 预算

创建名为 budget.bicep 的文件。此处显示的当前资源类型为 Microsoft.Consumption/budgets@2024-08-01。开始日期必须是某月第一天，并应在运行实验时调整。

```
targetScope = 'resourceGroup'

@description('预算资源名称')
param budgetName string

@description('账单货币表示的月度金额')
param amount int = 10

@description('通知收件人')
param contactEmail string

@description('UTC 表示的某月第一天')
param startDate string

resource monthlyBudget 'Microsoft.Consumption/budgets@2024-08-01' = {
  name: budgetName
  properties: {
    amount: amount
    category: 'Cost'
    timeGrain: 'Monthly'
    timePeriod: {
      startDate: startDate
    }
    notifications: {
      Actual_80_Percent: {
        enabled: true
        operator: 'GreaterThanOrEqualTo'
        threshold: 80
        thresholdType: 'Actual'
        contactEmails: [contactEmail]
```

```
        contactGroups: []
        contactRoles: []
      }
      Forecast_100_Percent: {
        enabled: true
        operator: 'GreaterThanOrEqualTo'
        threshold: 100
        thresholdType: 'Forecasted'
        contactEmails: [contactEmail]
        contactGroups: []
        contactRoles: []
      }
    }
  }
}

output budgetResourceId string = monthlyBudget.id
```

在资源组作用域部署模板。把示例开始日期替换为 UTC 表示的当前月或下个月第一天。

```
az deployment group create \
  --resource-group "$rg" \
  --name finops-budget-deployment \
  --template-file budget.bicep \
  --parameters \
      budgetName="$budget" \
      contactEmail="$owner_email" \
      startDate="2026-09-01T00:00:00Z" \
  --query "properties.outputs" -o jsonc
```

14.12.5 获取并验证预算资源

通过 Resource Manager 端点直接查询预算。这会捕获真实的控制平面表示，而不是依赖门户屏幕截图。

```
az rest \
  --method get \
  --url
"https://management.azure.com${scope}/providers/Microsoft.Consumption/budgets/${
budget}?api-version=2024-08-01" \
  --output jsonc
```

验证以下字段：资源 ID、作用域、金额、类别、时间粒度、开始日期、实际阈值、预测阈值、启用状态和通知收件人。不要把存在收件人当作邮件已发送的证明；通知需要后续评估和阈值条件。

14.12.6 查询当前月至今的成本

创建 query.json。此请求使用来源核验日期的当前 Azure 成本管理查询 API 版本。API 返回定义作用域的证据；当资源组没有已计费用量或成本数据尚未到达时，它可能不返回任何行。

```
cat > query.json <<'JSON'
{
  "type": "Usage",
  "timeframe": "MonthToDate",
  "dataset": {
    "granularity": "None",
    "aggregation": {
      "totalCost": {
        "name": "PreTaxCost",
        "function": "Sum"
      }
    }
  }
}
JSON

az rest \
  --method post \
  --url
"https://management.azure.com${scope}/providers/Microsoft.CostManagement/query?
api-version=2026-06-01" \
  --headers "Content-Type=application/json" \
  --body @query.json \
  --output jsonc
```

记录请求正文、响应列、响应行、返回的货币以及检索时间。零值或空结果对本实验而言都是有效证据。重要的是，查询契约能够标识作用域、期间、类型和聚合。

14.12.7 捕获部署与活动证据

```
az deployment group show \
  --resource-group "$rg" \
  --name finops-budget-deployment \
  --query "
{state:properties.provisioningState,timestamp:properties.timestamp,outputs:prope
rties.outputs}" \
  -o jsonc

az monitor activity-log list \
  --resource-group "$rg" \
```

```
--offset 1d \
--status Succeeded \
--max-events 50 \
-o jsonc
```

不同扩展资源与操作的活动日志覆盖情况可能有差异。Resource Manager 预算响应和部署记录是主要生命周期证据。如果活动日志查询没有结果，应记录这种缺失，而不是虚构事件。

14.12.8 说明控制行为

表 14-12 对预算证据的必需解释

问题	预期解释
预算达到 100% 时会停止资源吗?	不会。预算只评估支出，并路由通知或配置的操作。安全的运行时强制需要独立的工作负载特定控制。
告警会立即触发吗?	不一定。成本数据通常有延迟，预算评估也是周期性的。
预算金额是配额吗?	不是。它是财务跟踪目标；配额治理容量或服务限制。
增加配额会产生费用吗?	不会。成本随实际资源用量产生，但更高配额允许消费更多容量。
结果可以用于财务结账吗?	只有在所选成本语义和供应商数据已与发票或权威计量记录核对后才可以。

14.12.9 组装证据包

表 14-13 能力检查 C3 证据包

工件	必需内容
01-scope.json	订阅 ID、资源组 ID、标签、所有者、区域、捕获时间。
02-budget.bicep	受版本控制的预算定义。
03-deployment.json	部署状态、时间戳、输出预算资源 ID。
04-budget-resource.json	预算金额、期间、实际与预测通知、作用域、API 版本。
05-cost-query.json	作用域、请求正文、响应、货币/列、检索时间、新鲜度说明。
06-activity-log.json	相关生命周期证据，或明确说明未返回匹配事件。
07-interpretation.md	预算与配额的区别、预期评估延迟和安全响应策略。
08-cleanup.json	删除响应，以及资源组不再存在的证明。

14.12.10 可选配额扩展

使用 Azure Quotas 检查你已使用服务的相关配额，例如区域计算 vCPU 配额。记录当前用量、限制、作用域及配额是否可调。可以选择创建用量告警。不要只为实验申请增加配额。说明配额本身不会产生费用，预算和配额保护的是不同边界。

14.12.11 清理并验证删除

显式删除预算，然后删除资源组。即使删除作用域也会移除预算，显式删除仍使扩展资源的生命周期可见。

```
az rest \
  --method delete \
  --url
"https://management.azure.com${scope}/providers/Microsoft.Consumption/budgets/${
budget}?api-version=2024-08-01"

az group delete --name "$rg" --yes

az group exists --name "$rg"
# 预期最终值：false
```

确认预算 GET 请求现在返回未找到，并且资源组不再存在。本实验不需要任何可计费服务，但成本与账单记录可能依据供应商保留和财务记录要求继续保存。

14.12.12 能力检查的多云转换

在阿里云上，在定义的帐户或成本中心作用域创建成本预算或其他受支持预算，并捕获实际/预测告警配置。在腾讯云上，将预算管理与成本分配单元或基于标签的所有权视图结合。在火山引擎上，使用可用的账单中心预算或用量控制能力，并按需配合成本单元与配额中心。对于每个平台，都应保留相同的证据契约：作用域、所有者、金额或限制、期间、阈值语义、通知路径、数据延迟、API 或配置版本，以及清理。

注意：产品可用性、帐户权限、英文文档、预算行为和通知渠道都会变化。在书稿定稿前和每次运行实验前，应重新核验各个平台。

14.13 移交给 CloudDesk FinOps Agent

CloudDesk FinOps Agent 以后可以使用本章描述的托管能力，但本章不实现它的租户系统、Serverless 运行时、桌面审批、跨租户隔离或自主补救。可复用输出是一组小型、有界的能力契约。

```
finops.cost.read(scope, timeWindow, costSemantic, grouping, filters)
  -> values, currency, sourceTimestamp, freshness, queryEvidence

finops.budget.create(scope, amount, period, thresholds, notificationRoute)
  -> budgetResourceId, deployment Evidence, evaluationSemantics

finops.anomaly.detect(scope, baselinePolicy, evidenceWindow)
  -> anomaly, owner, likelyDrivers, confidence, investigationEvidence

finops.saving.verify(changeId, baseline, postChangeWindow, unitMetric,
sloPolicy)
  -> realizedSaving, variance, sideEffects, verificationEvidence
```

Agent 默认应只获得只读成本与资源查询能力。预算创建、承诺购买、配额变更、资源修改或自动关机需要独立身份和审批路径。每项输出都应引用作用域、时间窗口、成本语义、分配版本、新鲜度，以及支持性的资源或账单证据。即使云供应商只公开订阅级或帐户级控制，租户特定预算和速率策略仍必须在应用中强制执行。

14.14 关键要点

FinOps 是一种使技术价值最大化的运行框架和文化实践；降低成本只是一种可能结果，而非定义。

FinOps 控制平面视角把证据、所有权、价值、护栏、操作、验证和学习连接起来，而不集中每项工作负载决策。

云成本数据是延迟、派生且具有合同效力的证据。作用域、成本语义、货币、时间窗口、分配规则和新鲜度属于每项成本事实。

FOCUS 提供规范化的成本与用量契约，但供应商扩展、符合性缺口和核对仍应保持显式。

分配把共享供应商费用转化为可问责所有权。应结合并版本化层次、标签、成本中心和用量驱动因素。

预算、配额、额度、策略、承诺和异常信号保护不同边界。预算不是终止开关，配额也不是价格。

优化是对需求、用量、架构、放置、费率和运行决策的组合，并依据可靠性、安全、质量和产品结果进行评估。

单位经济性把技术支出连接到价值。AI 单位成本必须包含模型、检索、Agent 执行、安全、质量、运维，以及经过验证的结果分母。

Azure 通过成本管理、分配、预算、异常信号、导出、Advisor、承诺、配额、身份和活动证据公开参考 FinOps 控制平面。

潜在节省只有在受治理变更得到观测、依据需求规范化并对照成本与服务目标验证后，才转化为实现的节省。

多云可移植性来自保留 FinOps 证据与决策契约，而不是假定控制台标签等价。

AI Agent 可以凭借只读能力进行广泛分析，但财务或技术变更需要独立身份、审批、有界作用域、回滚和验证。

参考资料与官方文档

注意：来源说明：来源已于 2026 年 8 月 24 日核验。定价、折扣、交换/退款策略、产品名称、门户导航、API 版本、导出格式、FOCUS 符合性、计费延迟、预算权限、异常模型、配额、承诺资格和 AI 模型计量项，都必须在书稿定稿时以及每次运行实验前重新核查。

[1] FinOps Foundation. 什么是 FinOps？定义于 2026 年 3 月更新。

[2] FinOps Foundation. FinOps 框架概述：原则、作用域、角色、领域、能力和阶段。

[3] FinOps Foundation. FinOps 领域：理解用量与成本、量化业务价值、优化用量与成本，以及管理 FinOps 实践。

[4] FinOps Foundation. 能力：单位经济性。

[5] FinOps 开放成本与用量规范。FOCUS 规范 1.4。

[6] FinOps Foundation. 领域：量化业务价值。

[7] FinOps Foundation. 领域：优化用量与成本；能力：用量优化。

[8] Microsoft Learn. 教程：在 Microsoft 成本管理中创建和管理预算。

[9] Microsoft Learn. 账单与成本管理概述：异常检测、分配、Advisor 与告警。

[10] Microsoft Learn. Azure 架构良好的框架：收集与审查成本数据的架构策略。

[11] Microsoft Learn. Microsoft.Consumption/budgets@2024-08-01 Bicep 与 ARM 模板参考。

[12] Microsoft Learn. Azure 成本管理查询-用量 REST API，版本 2026-06-01。

[13] Microsoft Learn. Azure 成本管理导出 REST API 与 FOCUS 导出指南。

[14] Microsoft Learn. 使用成本告警监控用量与支出。

[15] Microsoft Learn. Azure Quotas 概述与配额用量告警。

[16] Microsoft Learn. 什么是 Azure 预留？以及预留与节省计划指南。

[17] Microsoft Learn. 规划和管理 Microsoft Foundry 成本。

[18] Microsoft Learn. Microsoft Foundry 模型配额与限制。

[19] Microsoft Learn. 管理并增加 Microsoft Foundry 资源配额。

[20] Microsoft Learn. Microsoft Foundry 模型的托管计算与预配吞吐量。

[21] 阿里云文档。架构框架：云成本管理框架。

[22] 阿里云文档。成本中心与共享成本分配。

[23] 阿里云文档。预算管理。

[24] 阿里云文档。成本分析与云成本异常检测。

[25] 阿里云文档。节省计划和利用率/覆盖率预算。

[26] 腾讯云文档。成本管理简介与成本分析。

[27] 腾讯云文档。预算管理。

[28] 腾讯云文档。成本分配单元、成本分配标签与分账单。

[29] 腾讯云文档。FOCUS 1.0 与 FOCUS 1.0 符合性缺口。

[30] 腾讯云文档。将账单保存到 COS 与导出中心。

[31] 腾讯云文档。FinOps 成本管理解决方案。

[32] 火山引擎文档。账单中心、成本单元与成本分析。

[33] 火山引擎文档。在 TOS 中配置账单数据存储及账单 API。

[34] 火山引擎文档。配额中心、配额用量与配额告警。

[35] 火山引擎文档。节省计划与服务计费模式。

第四篇 / 第 15 章

安全态势、合规与组织治理

将组织期望转化为可强制执行的资源状态与可审计证据

安全态势是组织意图、实际云状态、风险上下文、证据、例外与经验证补救之间的托管关系，而不只是仪表板上的分数。

本章内容

- 为何安全态势是期望控制与实际云状态之间持续管理的关系
- 组织层次如何决定治理控制的继承、作用域、所有权与证据
- 为何 RBAC、策略、资源锁、安全态势管理、审计与合规映射解决不同问题
- 策略即代码如何安全地从控制意图逐步演进到审计、有限推出、强制执行和评审
- 预防性、检测性、纠正性、补偿性与恢复控制如何协同工作
- 为何持续合规需要重复评估、漂移感知和证据新鲜度
- 为何云合规仪表板不等于法律认证或完整的控制评估
- CSPM 如何将业务上下文、暴露、漏洞和攻击路径优先级添加到配置发现中
- 如何治理例外、豁免、弃权、补偿控制和到期时间
- 如何在独立 Azure 实验中实施并验证拒绝策略
- 阿里云、腾讯云和火山引擎如何以不同的强制执行契约公开可比的治理意图
- CloudDesk FinOps Agent 如何复用治理检查、补救规划和态势解释能力

即使云安全仪表板上显示数百项绿色控制，组织仍可能无法回答一个基本问题：正在满足哪项业务要求，由哪项控制满足，在什么作用域内，以什么证据为依据，由谁授权，以及有效期有多长？仪表板只是一个视图。治理则是创建视图、指派所有者、约束变更、记录例外，并决定视图变红时应如何处理的系统。

因此，安全态势不是产品页面或一个分数，而是组织声明的控制意图与资源、身份、网络、数据路径、软件、AI 服务、依赖和运行流程实际状态之间的当前关系。每当资源被创建、角色被分配、公共端点被开放、策略被更新、模型部署被替换、新攻击路径出现或例外到期时，这种关系都会变化。

合规与此相关，但又有所不同。合规声明把外部或内部要求连接到控制目标、实现、评估、证据、所有权和风险决策。云平台可以自动化这条链路的重要部分：拒绝不允许的配置、评估现有资源、记录控制平面变更、呈现态势发现，以及补救选定状态。但云平台无法决定组织的法律义务、证明其无法观测的控制，或代表管理层接受剩余风险。

本章把安全态势和合规视为控制平面工程。Azure 通过管理组、Azure Policy、Activity Log、资源锁、补救任务、除外和 Defender for Cloud 提供参考模型。对阿里云、腾讯云和火山引擎的比较依据治理意图，而非产品名称的表面对称。核心问题始终相同：组织如何将一项期望转化为可强制执行、可评估、可解释且可逆的云控制？

本章论点

安全态势不只是仪表板。它是身份、网络、配置、加密、策略、审计证据、资源锁、态势发现、例外和补救工作流共同作用的结果。

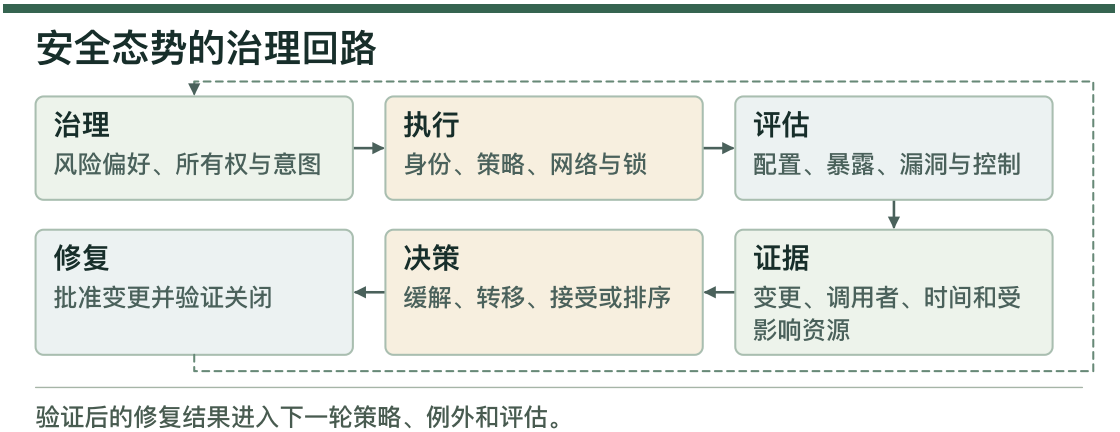


图 15-1 安全态势通过治理反馈回路来维持，该回路以经过验证的变更和重新评估结束。

15.1 安全态势是一种托管关系

有用的安全态势模型会比较两个持续演进的对象。第一个是预期控制配置文件：风险偏好、强制控制、获批架构、身份边界、数据处理规则、服务级别期望、例外策略与证据要求。第二个是观测到的资源集合：资源、配置、角色、公共暴露、漏洞、依赖、控制平面变更、数据位置与活动例外。安全态势就是两者之间经过评估的差异，并根据业务上下文和控制有效性进行调整。

该模型解释了为何单个百分比永远不够。两个组织即使拥有相同数量的错误配置，风险也可能大不相同，因为一项发现可能只暴露低价值沙盒，另一项却把面向互联网的身份连接到生产数据存储。如果态势系统只统计失败检查，却不考虑资源关键程度、可达性、特权、可利用性或业务影响，它可能给出精确分数，却导向糟糕决策。

安全态势还必须区分“缺少证据”和“存在无风险的证据”。资源看似合规，可能是因为它不在评估器清单中、供应商没有公开相关属性、评估已经陈旧、控制只能人工执行，或除外抑制了结果。可信的态势报告应随发现一并携带覆盖范围、新鲜度、作用域和例外元数据。

```
Security posture(t) = evaluate(intended controls, observed cloud state,
                               risk context, evidence quality, active
exceptions,
                               control effectiveness, and change history)
```

术语	首要问题	输出	它不能证明什么
治理	由谁决策、拥有、确定优先级、提供资金、评审和接受风险？	角色、策略意图、决策权、评审节奏	每项技术控制都已正确实施
策略	哪些资源状态或操作被允许、拒绝、修改或审计？	机器可评估的规则与分配	规则覆盖每项风险或法律要求
合规状态	观测到的资源在评估时是否符合已分配规则？	合规、不合规、除外、未知或等效的供应商状态	法律认证或完整的控制有效性
安全态势	配置、暴露、漏洞、身份、数据和依赖上下文会产生哪些风险？	按优先级排序的发现与建议	风险已被接受或补救
审计证据	谁在何时、何地更改了什么，结果如何？	控制平面事件及相关工件	最终状态安全或合规
控制评估	控制是否已实施、按预期运行，并产生期望结果？	包含证据和剩余缺口的评估结果	未来资源状态保持不变

安全态势具有时限

每项态势陈述都需要作用域和“截至”时间。没有新鲜度元数据的证据无法支持可靠的风险决策。

15.2 治理始于风险与决策权

NIST 网络安全框架 2.0 增加了“治理”（GOVERN）这一跨领域职能，用于建立并监控网络安全风险管理战略、期望与策略。这对云工程具有实际意义：配置规则不应源自孤立的控制台设置，而应追溯到组织目标、利益相关者期望、风险偏好、法律与合同要求，以及指定决策者。[1]

因此，云治理计划需要一条所有权链。高管或风险领导层设定容忍度和升级期望；安全架构把它们转化为控制目标；云平台团队实施可复用护栏、登陆区、策略定义和证据管道；工作负载所有者在护栏内设计，并承担应用特定风险；合规与内部审计独立评估证据；FinOps 和服务管理补充经济与运行后果。没有任何单一团队可以合理地承担每一个角色。

组织还必须决定哪些控制适用于全局，哪些取决于上下文。禁用匿名公共访问可以是全局基线；将部署限制到两个区域可能只适用于某类受监管数据；要求客户管理的加密密钥可能对一个工作负载必不可少，却对另一个造成不合理的运行负担。治理不是规则越多越好，而是以尽可能小且一致的控制集合可靠表达组织的风险决策。

角色	首要责任	不应悄然委派的决策
风险或高管领导层	设定风险偏好、升级阈值和可问责所有者	接受重大剩余风险
安全架构	把风险与要求转化为控制目标和获批模式	仅凭仪表板分数认证工作负载
云平台团队	实施层次、登陆区、可复用策略、证据和补救服务	对工作负载特定例外进行业务审批
工作负载所有者	分类数据、选择架构、运行控制、响应发现并为补救提供资金	无界例外，或把责任转移给供应商
合规/法律	解释适用义务并审查证据覆盖范围	提出缺少证据或服务契约支持的技术声明
内部审计/保证	独立评估设计和运行有效性	日常管理正在被评估的控制
FinOps/服务管理	把控制选择连接到成本、容量、服务价值与运行后果	仅因控制成本高昂就接受安全风险

没有所有者的控制就是等待发生的发现

每项已分配策略、态势建议、例外、补救任务和证据保留规则都应有可问责所有者与评审节奏。

15.3 组织层次是一种治理机制

云治理通过层次实现规模化。Azure 管理组提供高于订阅的作用域，使策略和访问分配可以继承到子作用域。阿里云资源目录、腾讯云组织和火山引擎企业组织提供相关的多帐户结构。层次不只是会计树，它还决定护栏附加在何处、哪些团队可以创建例外、如何聚合审计证据，以及委派管理员如何跨帐户运行。[15] [16] [26] [33] [41]

一个常见错误是原样复制人事组织结构图。业务单元经常变化，而安全、连接性、监管和平台要求通常更稳定。Azure 登陆区指南建议按共同的安全、合规、连接性和功能需要对工作负载分组，而不是仅为区域创建管理组。更普遍的原则适用于所有供应商：围绕治理继承与运行边界设计层次，再通过订阅、帐户、项目、资源组和标签映射业务所有权。[16]

高级作用域既带来能力，也带来危险。在接近根的位置分配拒绝策略可能阻止整个组织部署必需资源。宽松的子分配通常无法覆盖继承的显式拒绝。多帐户审计轨迹可以集中证据，但也会成为需要独立保护其目的地、保留和访问的特权资产。因此，根作用域变更比应用级变更需要更严格的评审、测试作用域、回滚计划和职责分离。



图 15-2 策略与控制意图向下继承，合规状态、发现和审计证据向上汇总。

层次层级	适当的治理内容	应避免
企业/根	最小化的全局拒绝边界、组织范围审计、身份先决条件、所有权要求	大型未经测试的策略包和工作负载特定设置
平台/登陆区	连接性、安全基线、区域与服务类别、共享监控、委派管理	在同一作用域中混合不兼容的工作负载风险类别
订阅/帐户/项目	环境、工作负载类别、预算、数据位置、服务所有权、平台变体	把单个订阅用作唯一的隔离与所有权机制
资源组/应用作用域	应用特定策略、资源锁、证据、补救和临时例外	没有父级可见性或到期时间的例外
单个资源	狭窄豁免、资源锁、诊断设置、关键资源保护	绕过 IaC 与中央证据的长期一次性配置

15.4 策略即代码：从意图到可强制执行状态

策略即代码把组织规则表达为有版本、可评审、可测试且可部署的工件。在 Azure Policy 中，定义描述条件与效果；分配把定义或计划绑定到作用域；参数调整可复用规则；除外记录受跟踪的例外；补救任务则可以纠正选定的现有资源。稳定抽象比任何一个供应商更广泛：控制目标、规则、作用域、效果、评估、证据、例外与补救。[6] [7] [8] [12] [13]

必须区分策略与授权。RBAC 或等效身份策略决定主体可以调用什么；资源策略约束哪些状态能够存在，或如何处理请求。用户可能拥有创建存储帐户的权限，却仍因请求资源缺少必需标签或启用公共访问而被拒绝。反过来，资源可能满足配置策略，而权限过大的身份仍可滥用它。两类控制都不可或缺。

策略也不同于资源锁。资源锁是一种针对作用域删除或修改的粗粒度控制，通常用于减少意外变更。它不能表达完整的期望配置、证明资源安全，也不能替代授权。态势管理又有所不同：它利用配置、漏洞、身份、暴露和依赖上下文发现风险并划分优先级。审计记录由谁执行了控制平面操作。这些机制彼此补充，而不能互换。

安全的策略生命周期始于影响分析。Microsoft 建议在强制执行前先采用审计效果，验证现有与新资源、检查误报、测试补救模板，并在广泛推出前使用有限作用域。同样的工程纪律也应应用到其他云的规则集。失效护栏造成的广泛停机可能超过它原本试图防止的错误配置。[9] [10]

控制平面机制	首要功能	典型输出	主要限制
身份与访问策略	为主体授予或约束操作	允许或拒绝的授权决策	不持续评估资源配置
资源配置策略	审计、拒绝、修改或要求资源状态	合规状态与强制执行结果	覆盖范围取决于受支持的资源属性和评估语义
服务控制策略/权限边界	限制成员帐户可用操作的上限	组织级操作边界	通常不会授予权限，也不能证明最终资源状态
资源锁/删除保护	防止或减少意外修改或删除	被阻止的控制平面操作	获授权管理员可以移除资源锁；数据平面行为可能不受影响
CSPM/态势评估	发现、结合上下文并划分安全风险优先级	发现、分数、建议、攻击路径	建议不是强制执行决策或风险接受
审计轨迹	记录管理事件和选定的数据平面事件	调用者、目标、操作、结果、时间、关联信息	操作证据不能证明最终状态安全
合规映射	把检查关联到控制或框架条款	覆盖视图与证据引用	自动检查只覆盖标准的一部分

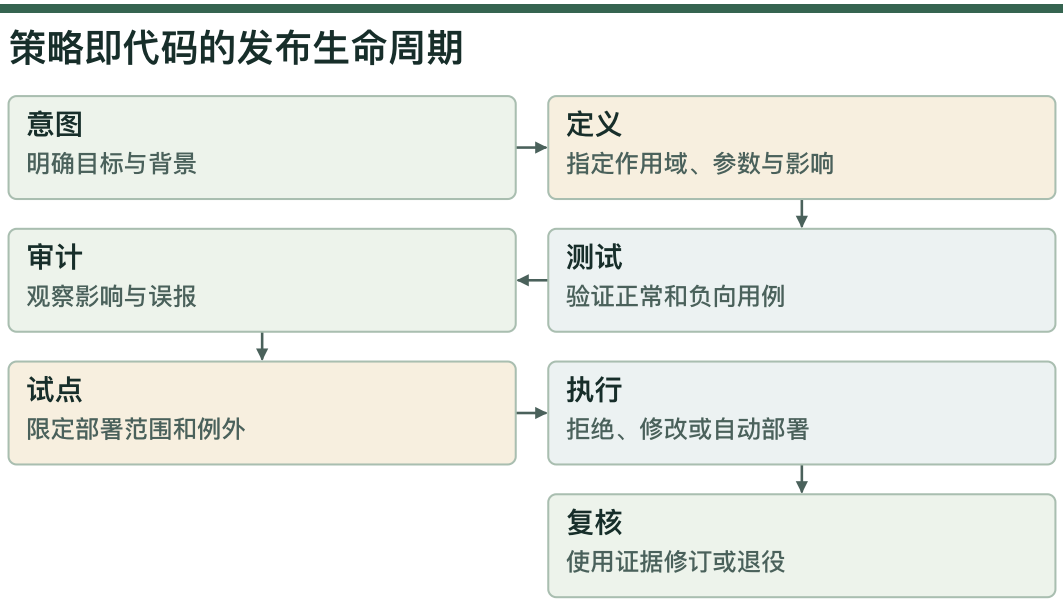


图 15-3 策略即代码需要自己的安全部署生命周期、证据、回滚和退役。

```
Policy artifact = control objective + rule + scope + effect + parameters
                  + version + owner + exception path + evidence contract
                  + safe deployment plan + retirement condition
```

15.5 控制类型协同构成系统

云控制通常分为预防性、检测性、纠正性、补偿性和恢复控制。这些类别描述控制的时机与功能，而不是重要程度。拒绝策略会阻止不允许的创建或更新请求；配置规则会检测现有状态；补救任务会修改不合规资源；除外加替代性保障可以充当补偿控制；当预防失效或纠正变更造成损害时，备份、回滚和恢复机制支持复原。

最强的设计通常组合多类控制。以公共对象存储为例：预防控制拒绝新的公共访问；检测规则查找已有的公共存储桶或帐户；纠正工作流依据获批计划关闭设置；审计轨迹记录谁更改了资源；在应用迁移期间，补偿控制可能临时限制源网络；如果变更破坏了合法集成，恢复控制会还原先前配置。

控制类型也会影响权限。预防控制在请求路径中执行并可能阻止交付，因此推出时需要严格测试。检测控制能够以较低运行风险广泛部署，但会产生对发现进行分流的义务。纠正自动化需要执行身份、最小权限、并发保护和回滚。补偿控制与例外需要显式的风险所有权和到期时间。恢复控制需要经过测试的工件，而不只是文档化意图。

控制类型	云实施模式	需保留的证据	失效方式
预防性	拒绝策略、服务控制策略、网络边界、准入规则、强制身份要求	被阻止的请求、策略与分配版本、作用域、调用者、关联 ID	误报阻止合法交付或应急响应
检测性	配置审计、CSPM 发现、清单查询、漏洞扫描、漂移检测	发现状态、资源快照、评估器版本、时间、覆盖范围	陈旧或不完整的清单产生虚假保证
纠正性	修改/部署补救、工作流、自动化运行手册、获批 IaC 变更	计划、身份、审批、变更结果、验证、回滚状态	自动修复造成停机或扩大访问范围
补偿性	替代性技术控制、受限例外、附加监控、人工评审	理由、控制等价性、所有者、到期时间、评审证据	临时权宜之计永久化并且不可见
恢复	回滚、还原、不可变备份、先前策略版本、紧急访问程序	恢复测试、工件完整性、恢复时间、决策记录	控制变更造成影响时，未经测试的恢复会失效

15.6 配置漂移与持续评估

配置漂移是获批或声明状态与观测状态之间的差异。漂移可能未经授权、由意外或紧急情况驱动、由供应商引入，或源于一次从未返回源代码控制的合法变更。态势系统不应假定每项差异都具有恶意，但必须使差异可见且可归因。

持续合规并不意味着每个属性每毫秒都接受评估。NIST 将持续监控描述为以足以支持风险决策的频率维持感知。云配置服务通常把事件驱动评估与定期扫描结合起来，结果可能滞后于刚刚完成的变更。因此，证据必须包含评估器、触发条件、观测时间、可用时的资源版本，以及下次预期扫描时间。[4]

事实来源至关重要。IaC 存储库可以表达预期状态，云控制平面则记录实际状态和管理事件。任何一个视图单独来看都不完整。IaC 可能省略平台或其他控制器应用的设置；控制平面可能显示最终状态，却不包含设计理由。成熟工作流会关联模板版本、部署记录、策略状态、配置历史和审计事件。

漂移管理还需要操作策略。有些差异应在创建前就被拒绝，有些应创建工单，有些可以自动修复，还有一些需要人工评审，因为最安全的变更取决于应用上下文。这种选择应写入控制定义与运行程序，而不是由第一个看到仪表板的人临时判断。

漂移问题	所需证据	可能响应
发生了什么变化?	变更前/后的配置或资源属性	把变更分类为预期、紧急、未经授权或供应商驱动

漂移问题	所需证据	可能响应
由谁或什么执行了变更?	调用者身份、服务角色、管道、API、源 IP、关联 ID	调查身份与变更路径
变更是否获批?	拉取请求、部署记录、工单、紧急审批	接受、回滚或升级处理
新状态是否违反策略?	当前策略版本、分配作用域、合规评估	拒绝未来变更、补救、设为除外或更新策略
会产生什么业务影响?	资源关键程度、暴露、依赖、数据分类、SLO	依据风险而非检查数量排序优先级
操作后的状态是否已验证?	最新评估、功能测试、遥测、活动日志	关闭、重新打开或启动恢复

15.7 合规证据不等于认证

监管或行业框架由结果、控制、程序、文档、组织责任和判断组成，其范围远超云资源属性。Azure Policy 监管计划和类似供应商包可以把自动检查映射到选定控制领域。它们是有价值的覆盖工具，但这种映射并不意味着通过自动检查就认证了组织或证明了每项控制。Microsoft 与腾讯云都明确区分自动检查覆盖和完整合规结论。[21] [35]

证据链应从要求开始，并保留每次转换。要求被解释为控制目标，目标通过一个或多个技术与程序性控制实施，评估程序通过检查、访谈或测试来考察实现，证据支持结果，责任机构则决定缺口需要补救、缓解、接受、转移还是报告。NIST SP 800-53A 强调评估程序，而不是把控制文本视为不证自明。[2] [3] [5]

资源被标记为合规，仍可能存在缺口。策略可能只评估一个属性，控制还可能要求文档化流程、员工培训、事件响应、合同条款、密钥托管或独立评审。相反，资源可能被标记为不合规，但一项已缓解除外记录了满足意图的替代控制。合规仪表板应公开这些区别，而不是将它们压缩为单一通过率。

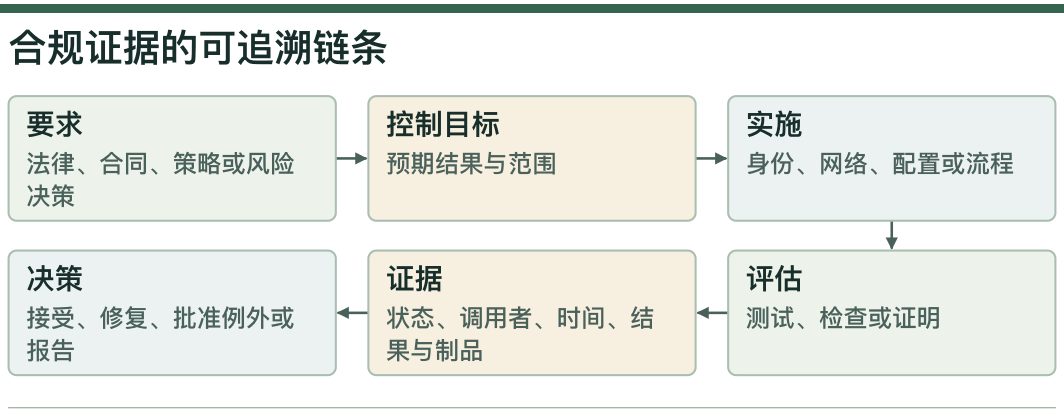


图 15-4 可辩护的合规声明会保留从要求到控制、评估、证据与决策的链路。

证据字段	重要原因	示例
要求标识符	防止自动规则与其业务或监管原因脱节	内部 SEC-12、合同条款、NIST 控制、数据驻留要求
控制目标	独立于单一供应商实现来陈述结果	生产存储不得允许匿名公共读取
实现	命名技术与程序性控制	拒绝策略、私有端点、所有者评审、公共访问告警
作用域与所有者	定义哪些资源和团队承担责任	生产订阅、数据平台所有者
评估程序	说明如何测试设计与运行	检查分配、尝试禁止部署、查询现有状态、审查日志
证据工件	支持可复现性与审计	策略 JSON、分配 ID、被拒绝的请求、合规结果、活动日志
例外/剩余缺口	防止无声抑制	豁免至迁移日期，同时采用补偿性防火墙规则
决策与评审日期	把证据连接到可问责的风险管理	由风险所有者接受；30 天后评审

合规免责声明

供应商合规仪表板和映射属于工程证据，而非法律建议、认证或完整控制评估的替代品。应与合格的法律、风险和保证专业人员共同确定适用义务。

15.8 CSPM 为配置发现添加风险上下文

配置合规询问资源是否匹配规则；云安全态势管理则提出更广泛的问题：哪种错误配置、身份、漏洞、数据敏感性、暴露和依赖组合构成最重要的风险？现代 CSPM 系统在配置检查之上添加资产清单、建议、风险优先级和攻击路径分析。Azure Defender for Cloud、阿

里云安全中心、腾讯云安全中心和火山引擎安全产品都公开了这类意图的不同实现。[19] [20] [30] [35] [36] [42]

这种区别对运维很重要。十条开放网络规则并不一定代表十项相等风险。一条可能暴露具有身份的生产主机上的管理端口，另一条可能只影响不含敏感数据的隔离测试实例。连接暴露、特权、可达资产、漏洞和业务关键程度的图，可以优先处理第一项，而不假装第二项可以接受。

CSPM 发现仍应进入同一治理回路。发现需要所有者、作用域、截止日期、证据、例外路径、补救计划和验证。分数和建议本身不会改变资源，也无法替代事件检测、工作负载保护、漏洞管理、数据安全或应用安全。态势管理是把这些信号连接到云资源上下文的决策支持层。

AI 服务扩大了态势界面。公共模型端点、权限过大的部署身份、未受保护的知识存储、未经评审的工具连接、敏感提示，以及跨资源攻击路径，都可能产生 AI 特定暴露，但不会改变底层治理模式。同样的控制问题仍然适用：哪项资源、哪个身份、哪条数据路径、哪项策略、哪些证据，以及哪位可问责所有者？

层	问题	代表性证据
配置合规	资源是否匹配特定规则？	策略状态、配置快照、规则版本
安全态势	结合上下文后，这项发现有多重要？	暴露、身份、漏洞、关键程度、数据、攻击路径
威胁检测	是否正在发生可疑或恶意活动？	告警、行为、指标、运行时事件
事件响应	哪些协调操作可以遏制、根除并恢复？	事件记录、时间线、遏制与恢复证据
保证	控制是否经过设计并随时间有效运行？	独立评估、样本、访谈、测试、例外、管理层决策

15.9 审计轨迹、资源锁与变更证据

Azure Activity Log 记录订阅级控制平面操作，例如资源创建、配置变更、访问策略变更和部署错误。阿里云 ActionTrail 记录控制台、API、SDK 和服务角色事件；腾讯云 CloudAudit 以及火山引擎审计与日志服务提供相应的管理事件证据。组织范围的轨迹可以把事件送到中央日志或对象存储目的地，以便延长保留期和进行分析。[17] [29] [32] [40]

审计证据应回答由谁、做了什么、何时、何地、如何执行以及结果如何。有用字段包括调用者身份、操作、资源 ID、时间戳、状态、来源、关联或请求 ID、策略决策和变更属性。当证据导出到受保护目的地，并具有保留、访问控制、完整性保障和交付失败监控时，证据会更有力。

资源锁解决的是更狭窄的问题。Azure 的 Delete 和 Read-only 锁保护订阅、资源组或资源免于意外删除或修改，并且针对这些操作可以覆盖普通用户权限。资源锁并非不可变安全边界：权限足够的管理员可以移除它，其行为主要位于控制平面。与 IAM、策略、审计和紧急访问程序结合时，资源锁对关键共享资源很有价值。[18]

审计、策略和资源锁应彼此关联。被拒绝的部署证明预防控制已经运行；活动日志说明调用者、目标、时间与失败；策略分配和定义解释请求为何被拒绝；合规的重新部署证明纠正后的请求满足规则。三者共同构成证据包，单个屏幕截图无法做到这一点。

证据来源	最适合回答的问题	保留/完整性关注点
策略状态	哪项规则以什么结果评估了哪个资源？	评估延迟、规则版本、除外状态、供应商覆盖范围
控制平面审计轨迹	谁在何时调用了哪个管理操作，状态如何？	导出、保留、特权访问、交付失败
配置历史	哪些属性或关系随时间发生了变化？	采样间隔、不受支持的资源、变更前后完整性
IaC 部署记录	哪个获批工件试图创建或改变状态？	存储库完整性、管道身份、参数与机密处理
CSPM 发现	结合上下文后，该状态为何具有风险？	评分模型变更、陈旧资产上下文、被抑制的发现
资源锁/删除保护	是否阻止了一次意外控制平面操作？	谁能移除资源锁，以及数据平面是否仍可变
评估或证明	哪些人工或独立证据支持该控制？	审阅者独立性、样本覆盖、到期与重新验证

15.10 例外、除外与风险接受

没有例外机制的治理系统会迫使团队采用隐藏绕过；拥有无界例外的治理系统则会把每项护栏变成可选建议。目标是受控的例外生命周期：明确作用域、所有者、理由、类别、补偿控制、到期时间、证据、审批与评审。

Azure Policy 区分排除与除外。被排除的作用域不接受分配评估；除外则保持为可见的除外状态，可以代表已缓解条件或豁免。已缓解除外表示策略意图通过其他方式得到满足；豁免则临时接受不合规状态。除外可以携带到期元数据，并在到期后继续保留记录以供追溯。[13] [14]

不同供应商的等效概念可能称为白名单、忽略发现、合规例外、规则排除或组织策略例外。语义问题比标签更重要。资源是否从评估中消失？是否仍然可见？例外能否自动到期？是否记录补偿控制？谁能创建它？该决策是否纳入组织级证据？

对于高风险控制，例外审批权限应与申请例外的团队分离。紧急破窗操作需要短期权限、事后评审和证据。反复出现的例外应触发架构评审：如果整个工作负载类别都无法满足某项控制，那么控制可能设计错误、作用域错误，或暴露了真实平台缺口。



图 15-5 每种不合规状态都应进入经过验证的补救路径，或显式且会到期的风险接受路径。

例外字段	必需内容
作用域	精确资源、资源层次、策略分配、控制和环境
类别	已缓解、豁免、临时运行例外、紧急或等效供应商类别
理由	为何当前无法满足正常控制，以及拒绝操作的业务后果
补偿控制	替代控制、监控、网络限制、审批或人工验证
风险所有者	获授权接受剩余风险的指定个人或角色
到期与评审	到期时间、评审节奏和提前撤销的触发条件
证据	发现、架构、测试、审批、关联工单和补救计划
退出计划	移除例外所需的具体操作、所有者、里程碑与验证

15.11 补救是一项受治理的变更工作流

补救不等于把仪表板变绿。正确的补救会保留服务意图、最小化变更风险、使用有界身份、记录审批、验证最终状态，并能在变更损害工作负载时逆转或补偿。因此，补救路径应复用本书其他部分确立的变更治理与可观测性纪律。

Azure Policy 补救任务可以使用附加到策略分配的托管身份，对现有资源应用修改操作或 DeployIfNotExists 模板。该身份只需要补救所需角色。新资源或更新资源可以在评估期间处理，而现有资源则需要显式补救任务。重要的架构要点不是具体效果名称，而是纠正自动化拥有自己的主体、权限、作用域、计划、证据和生命周期。[12]

自动补救应从低风险、可逆状态开始：添加必需元数据、启用诊断，或在测试后部署标准关联资源。关闭端点、更改加密密钥、替换身份模型或修改生产网络路由等高风险变更，需要工作负载感知的规划，并且通常需要人工审批。态势平台可以建议变更，但不应悄然获得无限权限来执行。

验证完成整个工作流。重新运行合规评估、检查资源状态、检查应用健康状况和 SLO、审查活动日志，并确认发现不会立即再次出现。应跟踪平均补救时间、重新打开率、误报率、例外年龄、控制覆盖和实际风险降低，而不只是已关闭建议数量。

补救阶段	所需控制
分流	验证作用域、新鲜度、严重性、业务关键程度和重复发现
规划	描述预期状态、受影响资源、依赖、爆炸半径、回滚与验证
授权	采用职责分离和基于风险的审批；将执行身份绑定到计划
执行	使用 IaC、策略补救或具备幂等性与并发保护的获批工作流
验证	重新评估合规、检查资源状态、审查遥测并确认服务行为
关闭	附加证据、记录剩余风险、移除临时权限并更新例外登记簿
学习	调整策略、测试用例、文档、所有权或架构，以减少再次发生

15.12 职责分离与治理运行模型

如果一个身份可以定义控制、在全局范围分配控制、将自身资源设为除外、执行补救、删除审计轨迹并批准结果，组织治理就会失效。职责分离不需要庞大的官僚机构，却确实要求在策略编写、策略部署、例外审批、补救执行和保证评审之间设置显式边界。

策略定义可以由平台存储库以代码方式管理。安全审阅者批准控制意图和测试用例；具有专用身份的部署管道把策略分配给有限作用域；工作负载所有者补救自身资源，或申请有时限的例外；独立的保证角色审查合规与审计证据。紧急路径可以使用，但必须进行事后评审。

运行模型应尽量减少永久高级权限。根作用域策略变更、组织控制策略、集中轨迹、补救身份和除外权限都是特权能力。在可用时使用即时访问、使用工作负载身份而非共享凭据、为策略代码设置受保护分支，并针对治理系统自身的变更配置告警。治理云的控制也是云中最敏感的资源之一。

治理活动	建议的主要执行者	独立检查
定义控制目标	安全架构/风险所有者	法律、合规、工作负载与平台评审
实施策略代码	云平台工程	自动测试与同行评审

治理活动	建议的主要执行者	独立检查
分配到广泛作用域	治理部署管道或获授权平台角色	变更审批与安全部署证据
申请例外	工作负载所有者	安全/风险审批者
执行补救	有界补救身份或工作负载团队	计划摘要、审批、遥测与变更后评估
审查证据	合规/保证/内部审计	源系统验证与抽样
更改审计保留期或目标位置	安全运营人员/平台所有者	独立告警和审批

对治理平面实施治理

策略定义、分配、除外、补救身份、审计目标位置和安全态势设置都是高价值云资源。应相应地对其进行清点、版本控制、监控和保护。

15.13 以 Azure 作为参考治理控制平面

Azure 明确呈现了治理对象模型。管理组用于组织订阅并提供继承作用域。策略定义用于表达规则；计划将相关定义组合成一个控制目标；分配把规则绑定到管理组、订阅、资源组或资源；排除和除外用于缩小评估范围；合规数据记录结果；补救任务纠正选定状态；Activity Log 记录控制平面操作；资源锁保护关键作用域；Defender for Cloud 则增加上下文态势分析和建议。[6] [11] [12] [15] [17] [18] [19] [22]

这些对象应被视为一张资源图。计划只有经过分配才会生效。分配若没有作用域和参数就没有意义。补救策略需要身份和角色。除外应引用相应分配，并带有所有者和到期时间。活动事件可以解释变更，但不能替代合规状态。Defender 建议可以帮助确定风险优先级，却无权更改资源。这张图就是治理系统。

Azure 模型还呈现了生命周期。内置策略定义和计划可能演进。自定义定义需要版本控制和退役机制。分配会被部署、更改、排除、除外和删除。合规评估具有触发条件和时延。补救任务会被创建并完成。锁会被添加和移除。审计数据会被保留或导出。治理架构应记录每种生命周期，而不是把“启用策略”视为一次性操作。

Azure 对象或服务	治理角色	生命周期问题
管理组	继承与委派的组织作用域	谁可以更改层次结构，订阅移动时哪些策略会随之移动?
策略定义	可复用规则与效果	如何测试、版本控制、弃用和退役?
计划	控制目标包	哪些定义和参数满足该目标?

Azure 对象或服务	治理角色	生命周期问题
策略分配	作用域、参数、强制模式和可选补救身份	如何限制、监控和更改推出范围，以及如何移除？
排除/除外	对评估或分配效果的受跟踪例外	谁批准了它、为什么批准，以及何时到期？
策略状态/合规数据	资源和分配的观测结果	结果有多新，缺少哪些覆盖？
补救任务	针对 modify 或 DeployIfNotExists 策略的纠正操作	由哪个身份和哪些角色执行，如何验证成功？
Activity Log	控制平面事件证据	导出到何处、保留多久、如何保护和关联？
资源锁	防止删除或修改的粗粒度保护	谁可以移除它，哪些操作仍可执行？
Defender for Cloud	态势、建议、图和攻击路径上下文	适用哪个计划、作用域、发现项所有者、截止日期和例外流程？

15.14 多云治理转换

本书讨论的四个云平台具有共同的治理意图：组织账户和资源、限定权限、记录管理事件、评估资源配置、汇聚证据，并确定安全风险的优先级。但它们并不提供完全相同的强制语义。Azure Policy 可以直接参与请求时拒绝、修改或相关资源部署。Alibaba Cloud Config、Tencent Cloud Config 和 Volcengine Config Audit 更强调持续清单、规则评估、合规包或基线、配置历史和证据；提供商特定的组织策略与权限边界可另行约束操作。[23] [31] [37]

这种区分可以避免错误等同。服务控制策略可以限制成员账户身份获准执行的操作，却不评估每个现有资源是否符合期望状态。配置规则可以把资源标记为不合规，而不阻止资源创建。态势检查可以建议补救，却不具备执行权限。标签策略可以规范标签操作，但成本分摊、授权和资源合规仍是彼此独立的契约。

多账户聚合是另一项共同意图。Alibaba Cloud 通过 Resource Directory 和委派管理，把 Cloud Config、ActionTrail 与 Security Center 集成起来。Tencent Cloud 提供组织级服务控制策略和标签策略，以及多账户 CloudAudit、Config 和 Cloud Security Center 能力。Volcengine 提供 Enterprise Organization 控制策略、Config Audit 中的账户组和跨账户合规基线、向 TLS 交付证据，以及多云安全态势能力。具体支持的服务、规则模板、补救方法和区域可用性都可能变化，因此实施前应加以验证。[27] [32] [39] [40] [41] [42]



图 15-6 。各提供商通过不同的组织、策略、配置、审计和态势契约，实现共同的治理意图。

工程意图	Azure	Alibaba Cloud	Tencent Cloud	Volcengine
组织账户与继承关系	管理组与订阅	Resource Directory、文件夹、成员账户	Tencent Cloud Organization、部门、成员	Enterprise Organization、部门、成员账户
限定组织权限上限	RBAC 加继承作用域上的策略	控制策略加 RAM	服务控制策略加 CAM	控制策略加 IAM
评估资源配置	Azure Policy 合规	Cloud Config 规则与合规包	Config 持续记录与评估	Config Audit 规则、基线、账户组
阻止不允许的状态	请求时执行拒绝及相关策略效果	使用控制策略、服务控制和受支持的预防/补救模式；按服务验证	使用组织权限边界和服务控制；Config 主要用于评估状态	使用组织控制策略和受支持的服务控制；Config Audit 主要用于评估状态
补救发现项	Modify/DeployIfNotExists 加补救任务和托管身份	在支持的情况下，使用 Cloud Config 模板或自定义补救	使用 CSC 建议及提供商/工作流补救；按规则验证 Config 能力	使用提供商特定的指导和工作流；不要假定存在与 Azure 等同的自动补救
记录管理事件	Activity Log 及导出的诊断目标位置	ActionTrail 到 SLS 或 OSS	CloudAudit 与集中式传输	配置历史并交付到 TLS；按资源验证操作审计服务
评估态势与上下文	Defender for Cloud CSPM 与攻击路径	Security Center CSPM	Cloud Security Center CSPM/CNAPP	Multicloud Security Platform 及相关安全产品
治理标签	Azure Policy 与标签继承/要求	Tag Policy 与 Cloud Config 必需标签检查	TCO 标签策略与 CAM 标签授权	项目/标签加配置规则；验证当前标签策略支持情况

记录非等价性

当中国云服务只评估却不阻止资源创建，或补救需要不同执行路径时，应把这种差异写入控制设计，而不要用产品映射掩盖它。

15.15 实验 4：强制执行资源策略并生成合规证据

本独立实验会创建一个自定义 Azure Policy：在专用资源组内，除非存储账户包含 owner 和 costCenter 两个所有权标签，否则拒绝创建。你将捕获一次被拒绝的部署、创建一个合规资源、触发评估扫描、查询合规状态、收集 Activity Log 证据，并删除所有实验资源。

本实验演示一种范围有限的控制平面能力，并不宣称标签是一项完整的安全控制。其目的是验证完整的治理路径：定义、分配、请求时强制、合规评估、证据、纠正后的部署和清理。

实验字段	值
能力目标	强制执行一项组织规则，并同时演示被拒绝的不合规和经验证的合规
Azure 资源	资源组、自定义策略定义、策略分配、一个合规存储账户
所需权限	有权在所选作用域创建策略定义和分配，并有权在实验资源组内创建和删除资源
成本边界	策略对象不需要专用计算资源；如果立即删除，存储账户产生的成本应可忽略不计。请核实当前价格和继承的控制。
证据	定义、分配、被拒绝的 CLI 输出、合规资源 ID、策略状态、Activity Log 导出、清理结果
清理契约	删除分配、自定义定义、存储账户和资源组；验证该资源组已不存在

实验安全

使用你获准创建自定义策略的订阅。不要在管理组或整个订阅作用域分配实验策略。将分配限制在专用实验资源组内。

步骤 1：定义名称并创建隔离作用域

```
SUBSCRIPTION_ID=$(az account show --query id -o tsv)
LOCATION=eastus
RG=book-lab-15-governance
POLICY_NAME=book-require-owner-costcenter
ASSIGNMENT_NAME=book-require-owner-costcenter-rg
SUFFIX=$(openssl rand -hex 4)
BAD_STORAGE=govbad${SUFFIX}
GOOD_STORAGE=govok${SUFFIX}
```

```
SCOPE=/subscriptions/${SUBSCRIPTION_ID}/resourceGroups/${RG}
mkdir -p evidence

az group create   --name "$RG"   --location "$LOCATION"   --tags owner=book-lab
costCenter=cloud-ai-paas purpose=governance-lab   > evidence/resource-group.json
```

存储账户名称必须全局唯一、采用小写字母，并符合提供商的长度等限制。如果所选区域不可用或受继承策略限制，请更改 LOCATION。继续前，将活动订阅和调用者身份保存到证据包中。

```
az account show > evidence/account.json
az ad signed-in-user show > evidence/caller.json 2>/dev/null || true
```

步骤 2：创建自定义拒绝规则

创建一个名为 policy-rule.json 的文件。该规则仅适用于存储账户，并在缺少任一必需标签时拒绝创建或更新。

```
{
  "if": {
    "allOf": [
      {
        "field": "type",
        "equals": "Microsoft.Storage/storageAccounts"
      },
      {
        "anyOf": [
          {
            "field": "tags['owner']",
            "exists": "false"
          },
          {
            "field": "tags['costCenter']",
            "exists": "false"
          }
        ]
      }
    ]
  },
  "then": {
    "effect": "deny"
  }
}
```

```
az policy definition create   --name "$POLICY_NAME"   --display-name "Book lab:
require owner and costCenter on storage accounts"   --description "Denies
storage accounts missing owner or costCenter tags."   --mode Indexed   --rules
@policy-rule.json   --subscription "$SUBSCRIPTION_ID"   > evidence/policy-
definition.json

POLICY_ID=$(az policy definition show   --name "$POLICY_NAME"   --subscription
"$SUBSCRIPTION_ID"   --query id -o tsv)
```

生产规则应包含元数据、所有者、版本控制、单元测试和安全部署记录。本实验把规则范围保持得很窄，以便轻松验证结果。

步骤 3：仅将规则分配给实验资源组

```
az policy assignment create   --name "$ASSIGNMENT_NAME"   --display-name "Book
lab: require storage ownership tags"   --scope "$SCOPE"   --policy "$POLICY_ID"
--description "Lab-only deny policy assigned to one resource group."   >
evidence/policy-assignment.json

ASSIGNMENT_ID=$(az policy assignment show   --name "$ASSIGNMENT_NAME"   --scope
"$SCOPE"   --query id -o tsv)
```

尝试任何部署之前，先确认分配作用域。如果 SCOPE 指向订阅而不是资源组，请先删除该分配并纠正作用域，再继续操作。

步骤 4：验证请求时强制执行

尝试创建一个不带必需标签的存储账户。该命令应因策略相关的拒绝而失败。请同时捕获标准输出和标准错误输出。

```
set +e
az storage account create   --name "$BAD_STORAGE"   --resource-group "$RG"   --
location "$LOCATION"   --sku Standard_LRS   --kind StorageV2   >
evidence/denied-deployment.stdout.json   2> evidence/denied-
deployment.stderr.txt
DENIED_EXIT=$?
set -e
printf "%s
" "$DENIED_EXIT" > evidence/denied-exit-code.txt

if [ "$DENIED_EXIT" -eq 0 ]; then
  echo "Expected policy denial, but deployment succeeded." >&2
  exit 1
fi
```

错误应标识策略定义或分配，并且通常包含 RequestDisallowedByPolicy 风格的结果。如果拒绝是由另一项继承策略造成的，请检查分配详细信息并记录干扰，而不要声称实验规则已经验证。[43]

步骤 5：创建合规资源

```
az storage account create   --name "$GOOD_STORAGE"   --resource-group "$RG"   --
location "$LOCATION"   --sku Standard_LRS   --kind StorageV2   --tags
owner=book-lab costCenter=cloud-ai-paas   > evidence/compliant-storage.json

az storage account show   --name "$GOOD_STORAGE"   --resource-group "$RG"   --
query
'{id:id,name:name,location:location,tags:tags,provisioningState:provisioningStat
e}'   > evidence/compliant-storage-summary.json
```

步骤 6：触发评估并收集策略状态

```
az policy state trigger-scan --resource-group "$RG"

az policy state list   --resource-group "$RG"   --query "[].
{resourceId:resourceId,complianceState:complianceState,policyAssignmentId:policy
AssignmentId,policyDefinitionName:policyDefinitionName,timestamp:timestamp}"   >
evidence/policy-state.json

az policy state summarize   --resource-group "$RG"   > evidence/policy-
summary.json
```

变更后，策略状态可能需要一段时间才会出现。请保留扫描时间，不要把立即得到的空响应视为合规证明。扫描完成后重新运行查询。对于此分配，合规存储账户不应显示为不合规。

步骤 7：收集控制平面审计证据

```
az monitor activity-log list   --resource-group "$RG"   --offset 2h   >
evidence/activity-log.json

az policy assignment show   --name "$ASSIGNMENT_NAME"   --scope "$SCOPE"   >
evidence/policy-assignment-final.json

az policy definition show   --name "$POLICY_NAME"   --subscription
"$SUBSCRIPTION_ID"   > evidence/policy-definition-final.json
```

证据包应使评审人员能够重建控制：定义、分配作用域、被拒绝的请求、纠正后的资源、当前策略状态、调用者、时间戳和 Activity Log 事件。截图可以作为证据包的补充，但不应取代机器可读工件。

验收测试	通过条件
作用域安全	分配 ID 位于专用实验资源组作用域之下
预防性控制	实验分配会拒绝创建或更新未加标签的存储账户
纠正后的部署	带标签的存储账户达到 Succeeded 状态
合规证据	评估后可以查询策略状态或摘要，且结果不会把合规账户报告为不合规
可审计性	证据包含调用者、时间戳、资源 ID、分配和定义 ID，以及 Activity Log 数据
可复现性	另一名获授权评审人员可以使用保存的工件和变量重新运行命令

步骤 8：移除治理对象并验证清理结果

```
az policy assignment delete   --name "$ASSIGNMENT_NAME"   --scope "$SCOPE"

az policy definition delete   --name "$POLICY_NAME"   --subscription
"$SUBSCRIPTION_ID"

az group delete   --name "$RG"   --yes

az group exists --name "$RG"   > evidence/resource-group-exists-after-
cleanup.txt
```

最终文件应包含 false。还要验证策略定义和分配均已不存在。仅保留经过净化的证据；不要在公共存储库中保留令牌、凭据或组织特定标识符。

提供商转换	可比的实施路径	重要的非等价性
Alibaba Cloud	Cloud Config 必需标签规则或合规包；Resource Directory 用于多账户作用域；ActionTrail 用于变更证据；在支持的情况下采用模板/自定义补救	配置规则可能只是评估现有资源，而不是拒绝最初的创建请求；预防和补救取决于服务和控制策略能力
Tencent Cloud	Config 规则和资源历史；TCO 标签策略或服务控制策略；CloudAudit；CSC CSPM 用于安全发现项	标签标准化、操作边界、配置评估和态势建议是彼此独立的控制契约

提供商转换	可比的实施路径	重要的非等价性
Volcengine	账户组中的 Config Audit 规则/基线；Enterprise Organization 控制策略；将配置和合规证据交付到 TLS	必须验证当前规则、预防、补救和标签策略支持情况；不要假定请求时拒绝能力与 Azure 等同

实验验收标准

只有当不合规请求被预期分配拒绝、纠正后的资源创建成功、控制平面证据可以复现，且清理过程同时移除了资源和治理对象时，实验才算通过。

CloudDesk 复用：治理与态势能力

CloudDesk FinOps Agent 可以把本章复用为三种不同能力。只读能力用于检查和解释治理状态。补救能力必须保持独立，要求使用范围受限的执行身份，并遵循 Agent Tools 章节建立的计划—批准—执行—验证契约。Agent 绝不能把合规映射转换为法律结论。

```
capability: governance.policy.check
risk_level: R0
mode: read_only
inputs:
  provider: string
  scope_id: string
  control_intent: string
outputs:
  assignments: array
  evaluated_resources: array
  non_compliant_resources: array
  exemptions: array
  evidence_freshness: object
  coverage_gaps: array
constraints:
  - preserve provider-native IDs and policy semantics
  - distinguish deny, evaluate, recommend, and remediate
  - report unknown or unsupported coverage explicitly
```

```
capability: governance.policy.remediate
risk_level: R3
mode: execute_after_approval
inputs:
  finding_id: string
  resource_ids: array
  approved_plan_digest: string
  rollback_plan: object
```

```
outputs:
  execution_id: string
  changed_resources: array
  verification_results: array
  residual_findings: array
  evidence_packet_uri: string
constraints:
  - least-privilege execution identity
  - no action without immutable approved plan
  - verify policy state, workload health, and audit trail after change
  - remove temporary authority when complete
```

```
capability: security.posture.explain
risk_level: R0
mode: read_only
inputs:
  finding_id: string
  business_context: object
outputs:
  affected_resources: array
  control_objectives: array
  risk_factors: array
  attack_path_context: array
  recommended_actions: array
  exception_status: object
  evidence_sources: array
  uncertainty: array
constraints:
  - explain prioritization without claiming certification
  - cite current source evidence and timestamps
  - separate provider recommendation from organizational risk decision
```

CloudDesk 用途	价值	强制护栏
FinOps 治理检查	在优化前检测缺失的所有权标签、预算作用域、公开暴露或不受支持的资源	只读；元数据缺失时不要推断所有者或成本中心
策略影响解释	解释部署或变更为何被拒绝，并识别起治理作用的分配	保留分配作用域和继承策略证据
补救提案	创建一份变更计划，把发现项、资源、期望状态、成本、SLO 和回滚联系起来	破坏性或安全敏感的变更需要人工批准
例外评审	查找已到期或理由不足的豁免，以及缺失的补偿性控制	未经责任人评审，不要撤销高影响例外

CloudDesk 用途	价值	强制护栏
态势摘要	根据业务上下文、暴露、身份、数据和攻击路径确定发现项优先级	评分仅供参考；风险接受应由获授权人员决定

本章小结

- 安全态势是组织意图、实际云状态、风险上下文、证据质量、例外和控制有效性之间具有时效性的关系。
- 治理确立决策权、所有权、风险偏好、策略意图、评审节奏和可接受的例外路径。
- 组织层次决定控制如何继承、证据如何汇聚，以及委派管理在何处运行。
- RBAC、资源策略、组织权限边界、资源锁、CSPM、审计跟踪和合规映射分别解决不同问题。
- 策略即代码需要测试、版本控制、安全推出、证据、回滚和退役，而不仅仅是 JSON 语法。
- 预防性、侦测性、纠正性、补偿性和恢复性控制应作为一个系统来设计。
- 持续合规需要重复评估、漂移感知、证据时效性和明确的操作规则。
- 提供商合规仪表板是有价值的工程证据，但并非法律认证或完整的控制评估。
- CSPM 为配置发现项添加暴露、漏洞、身份、数据、依赖关系和攻击路径上下文。
- 每项例外都需要作用域、所有者、理由、补偿性控制、到期时间、证据和退出计划。
- 补救是一项受治理的变更工作流，具有受限权限、验证和回滚机制。
- 多云治理必须转换意图、强制语义、评估时延、证据、例外行为和补救，而不能只转换产品名称。

参考资料与官方文档

产品名称、文档状态和引用的提供商说明已于 2026 年 8 月 25 日核查。内置策略、合规包、安全产品计划、区域可用性、认证、价格和补救能力都会变化。实施前请核实当前官方文档和适用的法律要求。

[1] National Institute of Standards and Technology。《NIST 网络安全框架（CSF）2.0》，NIST CSWP 29。https://csrc.nist.gov/pubs/cswp/29/the-nist-cybersecurity-framework-csf-20/final

[2] National Institute of Standards and Technology。《信息系统与组织的安全和隐私控制》，SP 800-53 Rev. 5。https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final

[3] National Institute of Standards and Technology。《评估信息系统与组织中的安全和隐私控制》，SP 800-53A Rev. 5。https://csrc.nist.gov/pubs/sp/800/53/a/r5/final

[4] National Institute of Standards and Technology。《信息安全持续监控（ISCM）》，SP 800-137。https://csrc.nist.gov/pubs/sp/800/137/final

[5] National Institute of Standards and Technology。风险管理框架。https://csrc.nist.gov/projects/risk-management/about-rmf

[6] Microsoft。Azure Policy 概述。https://learn.microsoft.com/en-us/azure/governance/policy/overview

[7] Microsoft。Azure Policy 定义结构基础与策略规则。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/definition-structure-basics 和 https://learn.microsoft.com/en-us/azure/governance/policy/concepts/definition-structure-policy-rule

[8] Microsoft。Azure Policy 效果基础。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effect-basics

[9] Microsoft。设计 Azure Policy as Code 工作流。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-as-code

[10] Microsoft。安全部署 Azure Policy 分配。https://learn.microsoft.com/en-us/azure/governance/policy/how-to/policy-safe-deployment-practices

[11] Microsoft。获取 Azure 资源的合规数据。https://learn.microsoft.com/en-us/azure/governance/policy/how-to/get-compliance-data

[12] Microsoft。使用 Azure Policy 补救不合规资源。https://learn.microsoft.com/en-us/azure/governance/policy/how-to/remediate-resources

[13] Microsoft。Azure Policy 除外结构。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/exemption-structure

[14] Microsoft。理解 Azure Policy 中的作用域。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/scope

[15] Microsoft。使用 Azure 管理组组织资源。https://learn.microsoft.com/en-us/azure/governance/management-groups/overview

[16] Microsoft。Azure 登陆区的 Cloud Adoption Framework 管理组设计。https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/design-area/resource-org-management-groups

[17] Microsoft。Azure Monitor 中的 Activity Log。https://learn.microsoft.com/en-us/azure/azure-monitor/platform/activity-log

[18] Microsoft。锁定 Azure 资源以保护基础设施。https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/lock-resources

[19] Microsoft。Defender for Cloud 中的 Cloud Security Posture Management。https://learn.microsoft.com/en-us/azure/defender-for-cloud/concept-cloud-security-posture-management

[20] Microsoft。Defender for Cloud 中的 Security Explorer 与攻击路径。https://learn.microsoft.com/en-us/azure/defender-for-cloud/concept-attack-path

[21] Microsoft。Azure Policy 计划定义中的法规合规。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/regulatory-compliance

[22] Microsoft。Azure Policy 计划定义结构。https://learn.microsoft.com/en-us/azure/governance/policy/concepts/initiative-definition-structure

[23] Alibaba Cloud。什么是 Cloud Config？https://www.alibabacloud.com/help/en/cloud-config/latest/what-is-cloud-config

[24] Alibaba Cloud。合规包概述。https://www.alibabacloud.com/help/en/cloud-config/latest/compliance-packages-overview

[25] Alibaba Cloud。Cloud Config 补救概述。https://www.alibabacloud.com/help/en/cloud-config/latest/remediation-settings-overview

[26] Alibaba Cloud。什么是 Resource Directory？https://www.alibabacloud.com/help/en/resource-management/resource-directory/product-overview/resource-directory-overview

[27] Alibaba Cloud。Resource Directory 可信服务。https://www.alibabacloud.com/help/en/resource-management/resource-directory/user-guide/overview-1

[28] Alibaba Cloud。标签策略概述。https://www.alibabacloud.com/help/en/resource-management/tag/user-guide/overview

[29] Alibaba Cloud。什么是 ActionTrail？https://www.alibabacloud.com/help/en/actiontrail/product-overview/what-is-actiontrail

[30] Alibaba Cloud。Security Center 中的 Cloud Security Posture Management。https://www.alibabacloud.com/help/en/security-center/user-guide/cspm

[31] Tencent Cloud。Config 概述。https://www.tencentcloud.com/document/product/1164/51489

[32] Tencent Cloud。使用 CloudAudit 进行多账户日志收集与审计。https://www.tencentcloud.com/document/product/1220/57831

[33] Tencent Cloud。Tencent Cloud Organization 中的服务控制策略。https://www.tencentcloud.com/document/product/1031/51869

[34] Tencent Cloud。创建标签策略。https://www.tencentcloud.com/document/product/1031/55962

[35] Tencent Cloud。Cloud Security Center 中的云资源配置检查。https://www.tencentcloud.com/document/product/1008/80336

[36] Tencent Cloud。Cloud Security Center 产品概述。https://www.tencentcloud.com/document/product/1008/79817

[37] Volcengine。Config Audit 产品介绍。https://www.volcengine.com/docs/87053/1999420

[38] Volcengine。Config Audit 规则策略。https://www.volcengine.com/docs/87053/2407063

[39] Volcengine。Config Audit 账户组与跨账户合规基线。https://www.volcengine.com/docs/87053/2024041

[40] Volcengine。将 Config Audit 证据交付到 TLS。https://www.volcengine.com/docs/87053/2024032

[41] Volcengine。Enterprise Organization 控制策略概述。https://www.volcengine.com/docs/6723/147486

[42] Volcengine。Multicloud Security Platform 合规结果。https://www.volcengine.com/docs/69190/1282941

[43] Microsoft。排查 RequestDisallowedByPolicy。https://learn.microsoft.com/en-us/azure/azure-resource-manager/troubleshooting/error-policy-requestdisallowedbypolicy

[44] Microsoft。Azure CLI policy state 命令参考。https://learn.microsoft.com/en-us/cli/azure/policy/state?view=azure-cli-latest

AI PaaS 控制平面

当模型、知识和智能体都成为托管资源，一次正确的 API 响应只说明了生命周期的一小部分。这四章从数据与模型平台出发，逐步核对资源版本、访问边界、评估、运行证据和退役条件。

16 从数据与 AI 平台和机器学习 PaaS 到 AI PaaS

17 从模型 API 到 AI PaaS 资源

18 RAG 不是向量数据库

19 作为受治理云资源的智能体平台

第五篇 / 第 16 章

从数据与 AI 平台和机器学习 PaaS 到 AI PaaS

云控制平面如何从模型生命周期扩展到模型、知识、工具、评估与智能体

AI PaaS 并非始于云提供商加入模型目录之时。它是在托管数据与机器学习生命周期扩展为将模型服务、知识资源、评估、工具和智能体作为生产级云对象进行治理之后逐步形成的。

本章内容

- 为何数据服务仍是传统机器学习与生成式 AI 系统的共同基础
- 传统机器学习 PaaS 如何把数据资产、实验、训练作业、模型、端点和监控转化为托管资源
- 为何数据与 AI 平台不只是把笔记本、数据引擎和 GPU 集群放在一起
- 生成式 AI 改变了控制平面的哪些部分，又保留了哪些部分
- AI PaaS 如何把模型服务、知识、提示词、工具、评估、安全策略和智能体加入托管资源图
- 为何 MLOps、LLMOps 与 AgentOps 相互重叠，而不是彼此取代
- Azure、阿里云、腾讯云与火山引擎的平台路径如何以不同产品边界表达相近的工程意图
- 如何在选择云提供商之前，使用架构工作表分离数据、模型、知识、工具和智能体资源
- 哪些反模式会把 AI 平台变成昂贵且彼此割裂的服务集合
- CloudDesk FinOps Agent 如何复用平台清单、血缘和生命周期缺口评估能力，而不在本章中实现这些能力

“AI 平台”一词常被当作一个稳定的产品类别。实际上，它可能指笔记本服务、分布式训练调度器、模型注册表、托管推理 API、数据湖仓、提示词试验场、检索系统或智能体构建器。营销页面或许会把这些能力放进同一张图，但工程团队仍须判断：哪些对象是持久的、由谁负责、依赖哪些数据、如何发布、产生哪些证据，以及如何退役。

这一历史演进十分重要。云提供商最先将机器学习生命周期产品化：托管开发环境、弹性训练作业、实验跟踪、数据与环境资产、模型注册表、流水线、端点和监控。这就是机器学习 PaaS。它不只是提供 GPU，而是把反复出现的机器学习工程责任纳入控制平面，使团队能够复现、授权、调度、观测和审计过去依靠脚本与个人计算机拼接起来的工作。

生成式 AI 改变了重心，却没有消除对这一基础的需要。如今，许多团队使用预训练模型，而非从头训练每个模型。生产系统依赖模型目录、托管推理、提示词与应用配置、检索增强生成、评估套件、安全策略、工具、记忆和智能体运行时。这些是新增的资源家族，使平台从“管理模型生命周期”扩展为“管理模型支持的应用与智能体生命周期”。

本章说明控制平面的这一扩展。首先讨论数据基础，因为模型、检索系统和智能体无法脱离其转换或引用的数据而独立治理。随后形式化机器学习 PaaS 的资源模型，识别 AI PaaS 引入的新对象，并映射四家云提供商的路径。Azure 通过 Azure Databricks、Azure Machine Learning 和 Microsoft Foundry 提供主要教学参照；阿里云、腾讯云与火山引擎则按工程意图比较，而不强求产品对称。

本章论点

机器学习 PaaS 将数据、训练、模型和部署的生命周期产品化。AI PaaS 保留这一基础，并加入受治理的模型服务、知识、提示词与应用配置、工具、评估、安全控制和智能体资源。



图 16-1 生产级 AI 系统同时建立在托管数据基础和托管模型基础之上。

16.1 AI 平台的演进始于数据，而非模型

只提供模型训练或推理、却没有稳定数据系统的平台，只能算不完整的 AI 平台。训练数据必须可发现、可分类、可清洗、可版本化、可转换，并能关联到模型运行。检索数据必须经过解析、分块、增强、授权、刷新，并可追溯至来源。智能体工具依赖具有业务语义、所有权和访问边界的运营数据。即使是托管模型 API，也会接收上下文、产生追踪，并参与应用工作流，而这些工作流的正确性依赖数据契约。

因此，平台的数据侧要回答模型侧无法回答的问题：某个特征源自何处？哪项转换生成了某张表？检索到的段落来自哪个文档版本？哪些用户有资格查看？哪条流水线刷新了它？适用什么保留规则？架构变更会影响哪些下游模型、索引、报表或智能体？数据血缘不是装饰性元数据，而是调试、影响分析、合规证据和受控变更所需的依赖图。

传统机器学习中的特征工程很好地说明了这一点。原始字段会被清洗、编码、组合、按时间窗口聚合并筛选。托管特征平台为此加入定义、版本、计算逻辑、在线与离线服务、新鲜度预期和复用机制。工程目标不是存储更多列，而是防止训练与服务偏差，并让特征成为受治理资产，而不是在各个笔记本中复制的无文档表达式。

生成式 AI 改变了转换方式，却没有改变治理责任。文档语料可能被转换为分块、嵌入、元数据和资格筛选条件；一段对话可能产生反馈、评估用例、安全事件和成本记录；一次工具调用可能读取或修改运营系统。即使用户只看到聊天回复，这些流也属于数据资产范围。若数据血缘止于模型 API，AI 系统最关键的部分就会不可见。

因此，本书所依据的课程材料把传统 AI PaaS 描述为“两条腿”：一条是数据开发与治理平台，另一条是用于训练、评估、部署和监控的模型平台。现代 AI PaaS 又加入模型服务、检索、工具和智能体，但这两条腿的结构依然存在。不同提供商的产品封装可能不同，工程责任却仍然存在。

数据责任	传统机器学习示例	生成式 AI / 智能体示例	控制平面证据
获取与准入	已批准的训练数据集	已批准的文档源或工具数据源	来源登记、所有者、区域、分类
转换	清洗、特征计算、标注	解析、分块、嵌入、脱敏、增强	流水线版本、代码或组件摘要、运行记录
质量	缺失值、漂移、标签质量	提取失败、过期分块、引用缺口、受污染内容	质量规则、失败记录、评估结果
血缘	数据集到运行再到模型	来源到分块、检索再到答案	依赖图、追踪标识符和来源标识符
授权	工作区与数据访问	文档资格、租户筛选、工具作用域	角色分配、策略、查询筛选器、访问决策
保留与退役	删除过时数据集和特征	删除过期索引、嵌入、对话和记忆	保留策略、删除事件、清除证据

数据是主动依赖项

模型层无法弥补缺失的所有权、过期的血缘、不一致的业务定义，或从未附加到数据的授权规则。

16.2 机器学习 PaaS 产品化了什么

在机器学习 PaaS 出现之前，典型的模型项目会组合本地笔记本、手工预配的 GPU 计算机、临时存储路径、复制的环境、训练脚本、记录指标的电子表格，以及用于部署的自定义 Web 服务。模型制品往往是唯一被视为持久对象的东西，其生成过程全部隐含不明，而生产责任的介入又太晚。

机器学习 PaaS 改变了管理单元。工作区或项目成为协作与授权作用域；数据和环境成为版本化资产；训练成为具有声明式输入、计算、代码、参数、输出、日志与状态的作业；实验对可比运行进行分组；流水线表示可重复组件之间的依赖；注册表存储带版本和来源的模型；端点将稳定的调用契约与一个或多个部署分离；监控则把生产行为重新连接到模型生命周期。

Azure Machine Learning 直接描述了这一目标：它管理机器学习项目生命周期，支持训练和部署，并提供 MLOps，以便在安全、可审计的生产环境中进行监控、再训练与重新部署。其工作区组织共享资产和协作；流水线自动执行工作；托管端点抽象在线与批量推理；作业制品与血缘则支持可审计性。[4]

关键变化并不是笔记本、调度器或注册表的存在，这些组件都可以单独安装；变化在于它们之间持久的关系。平台能够回答：哪些数据、代码、环境、计算、参数和运行产生了某个模型；哪个部署提供该模型；哪个身份更改了端点；以及哪些证据支持晋级或回滚。这就是机器学习 PaaS 的控制平面价值。



图 16-2 机器学习 PaaS 将持久资产与策略同临时作业和运行时执行分离。

机器学习 PaaS 对象	持久契约	典型证据	缺失时的常见故障
工作区 / 项目	所有权、成员、策略、配额、资产命名空间	角色分配、设置、活动历史	缺乏稳定的协作或责任边界
数据资产	版本化引用、架构、位置、分类	血缘、数据画像、访问记录	训练无法复现或审计
环境 / 镜像	运行时依赖项与执行契约	镜像摘要、软件包锁定、构建来源	“在我的机器上能运行”演变为生产缺陷
计算 / 配额	容量系列、加速器、缩放与分配边界	分配记录、利用率、作业放置	成本失控或训练容量不可用
作业 / 运行	代码、输入、参数、环境、输出、状态	日志、指标、制品、时间戳	结果无法比较或追溯
流水线 / 组件	可复用步骤、依赖项、计划与接口	运行图、组件版本、重试记录	胶水代码和人工交接主导运维
模型资产 / 注册表	版本、格式、指标、来源和批准状态	模型卡、血缘、签名、评估记录	文件被误当作生产模型发布
端点 / 部署	调用契约、模型绑定、流量、缩放和回滚	部署修订版、请求指标、变更日志	使用方依赖可变实现
监控器	质量、漂移、延迟、失败和成本预期	信号、警报、调查和再训练证据	生产性能下降由用户首先发现

16.3 数据与 AI 平台：一个资产范围，多种生命周期

“数据与 AI 平台”不只是同一门户里同时提供 SQL 和 Python。连贯的数据与 AI 平台把企业数据存储、摄取、转换、治理、分析、特征计算、模型开发、部署和运营反馈连接起来，同时不抹平这些对象各自不同的生命周期。

Azure Databricks 是以数据为中心路径的典型例子。Microsoft 将其描述为面向企业数据、分析与 AI 的统一开放式分析平台。其湖仓模型为工程师、科学家、分析师和生产系统提供共享数据基础，而 Apache Spark、Delta Lake、MLflow 与目录服务等技术则支持数据处理和模型工作。[3] Azure Machine Learning 又为工作区、作业、资产、注册表、托管端点和 MLOps 提供专用机器学习控制平面。两个平台在开发和模型运维的某些方面存在重叠，但其资源模型和运营边界仍可区分。

阿里云遵循相似的分层逻辑，但产品边界不同。DataWorks 跨 MaxCompute、E-MapReduce、Hologres、Flink、AnalyticDB 和 PAI 等引擎，集成数据摄取、开发、治理与分析。[11] PAI 通过 DSW 开发环境、DLC 训练作业、Designer 工作流、Model

Gallery、模型管理和 EAS 推理服务提供机器学习生命周期。[12] DataWorks 可以调度 PAI 节点，使数据转换以及训练或推理工作参与同一编排链。其价值并非形成单体产品，而是让数据生命周期与模型生命周期实现明确集成。

腾讯云 TI-ONE 为 AI 工程师提供一站式训练与推理环境，腾讯云数据服务则提供外围的数据工程基础。火山引擎通过 VeDI 和 DataAgent 提供数据智能路径，通过 Ark 提供模型路径，并通过 AgentKit 提供智能体运行时路径。这些例子展示了反复出现的架构：数据资产范围、模型生命周期以及应用或智能体生命周期相互连接，但不应被压缩成一个不加区分的平台框。

有效的架构会追问每种生命周期在哪里接受治理。数据工程师可能负责转换与质量；机器学习工程师可能负责训练、模型验证和服务；应用团队可能负责提示词、检索策略、工具契约、用户体验与发布；平台团队可能负责共享身份、网络、可观测性、配额和已批准模式。平台应通过资源和证据连接这些角色，而不是假装一个用户界面就能消除它们。

层	主要托管对象	主导生命周期	典型所有者
数据平台	来源、表、文件、流、目录、质量规则、血缘、计划	准入、转换、验证、发布、保留	数据工程 / 治理
机器学习 PaaS	工作区、数据资产、环境、作业、流水线、模型、端点	实验、训练、评估、注册、部署、监控、再训练	数据科学 / 机器学习工程
AI PaaS	模型服务、提示词、知识、评估、安全策略、工具、智能体	组合、评估、发布、观测、修订、退役	AI 应用 / 平台工程
业务应用	用户工作流、事务、审批、记录与结果	发布、运营、支持、改进	产品与应用团队

16.4 生成式 AI 改变了什么，又保留了什么

生成式 AI 降低了为每个应用训练任务专用模型的需求。团队可以选择预训练模型，通过托管服务调用，并借助提示词、检索、工具、结构化输出和评估改进应用。这是开发者工作流的深刻变化，把大量应用特定行为上移到模型权重之上。

这一变化创造了新对象。模型目录必须描述提供商、能力、版本、安全条款、区域、价格和部署选项；模型部署必须把模型版本与容量、身份、网络、安全和遥测绑定；提示词与应用配置必须版本化并接受测试；知识源、摄取流水线、索引和访问规则必须受治理；评估需要数据集、指标、评分器、阈值和运行证据；工具需要架构、身份、风险等级、批准和审计跟踪；智能体又加入定义、运行时、会话、追踪、记忆和发布状态。

生成式 AI 没有消除的内容同样重要。数据仍须具备质量、血缘、访问控制、保留与新鲜度；生产变更仍须具备发布门禁、回滚、可观测性和成本归因；微调或自托管模型仍需要训练、模型管理和部署运维。即使不进行训练，应用仍需要可审计、可复现的生命周期。现在必须复现的对象可能是模型部署、提示词版本、知识发布、工具策略和智能体版本的组合，而非单一模型文件。

组织重心也随之转移。传统机器学习项目通常旨在创建并运营模型；生成式 AI 项目则经常运营许多共享模型、但在知识、策略、提示词、工具和用户方面各不相同的应用。因此，平台必须分配并治理可复用模型服务，同时保留应用级隔离和证据。这就是 MaaS 必要但不充分的原因：它解决托管模型访问，却没有解决 AI 应用的完整生命周期。

机器学习 PaaS 的延续性	AI PaaS 的新重点	为何两者都不可或缺
版本化数据与血缘	知识源、分块、引用和对话证据	模型支持的答案仍依赖受治理的数据转换
实验跟踪	提示词、检索、工具和智能体评估运行	即使模型权重不变，行为也可能变化
模型注册表与部署	模型目录、MaaS 部署与路由	团队可能使用、微调或托管不同模型系列
监控与漂移	追踪、安全、依据性、工具结果和令牌经济性	端点即使正确，也可能产生不安全或缺乏依据的答案
流水线与发布门禁	知识刷新、评估门禁与智能体发布	应用行为涉及的范围超出训练流水线
身份与网络控制	文档资格、工具作用域与智能体执行身份	授权路径变得更长、更动态

运营单元已经改变

模型仍然重要，但生产发布越来越成为模型、提示词、知识、工具、评估、安全与智能体资源的复合体。

16.5 AI PaaS 资源模型

判断 AI PaaS 的标准应是其公开的资源和生命周期责任，而不是门户里是否有聊天试验场。面向生产的平台至少需要一个管理作用域、一个或多个模型服务契约，以及管理应用行为、知识与工具资源、评估证据、可观测性、策略和经济性的方式。

具体封装方式各异。Microsoft Foundry 目前把智能体、模型和工具统一到同一管理分组下，并通过同一个 Azure 资源提供程序命名空间应用基于角色的访问控制、网络和策略，同时在项目级提供追踪、监控与评估。[5] 阿里云 Model Studio 强调通过通义千问和第三方模型提供一站式模型服务，支持与 OpenAI 兼容的 API 及应用能力。[15] 腾讯云把 TI-

ONE 的训练与推理同 ADP 的应用和智能体构建分开；火山引擎把 Ark 模型服务同 AgentKit 运行时及更广泛的 VeDI 数据智能生态分开。产品边界不同，但资源责任仍可比较。

有用的概念模型把 AI PaaS 视为控制平面的扩展。机器学习 PaaS 资源继续服务于训练、微调、自定义模型、批量评分和 MLOps；AI PaaS 则加入用于使用模型并将其组合为应用的资源。并非每家提供商都把每种资源实现为一等对象，有些能力仍可能处于预览、受区域限制，或嵌入应用构建器。架构师应记录这种不等价，而不是凭假设填补空白。



图 16-3 AI PaaS 扩展资源图，同时保留下层的数据与机器学习生命周期。

AI PaaS 资源家族	控制平面状态	数据平面操作	发布证据
模型目录与模型卡	提供商、版本、能力、条款、区域、基准、退役状态	发现并选择模型	选择依据与已接受约束
模型部署 / 端点	模型绑定、容量、网络、身份、安全、路由	调用推理	部署记录、调用遥测与成本
提示词或应用配置	指令、变量、响应架构、版本、所有者	构造请求并解释响应	评估结果与晋级版本
知识资源	来源、摄取、架构、索引、访问策略、发布	检索依据与引用	新鲜度、授权与检索评估
工具 / 连接器	架构、目标、身份、作用域、风险、批准、超时	读取或修改外部系统	计划、决策、调用与结果证据
评估套件与运行	数据集、指标、评分器、阈值、环境	执行质量与安全测试	不可变结果与发布决策
安全策略	筛选器、分类器、阈值、例外与升级	检查或阻止内容和操作	策略版本、决策与事件证据
智能体定义与运行时	模型、指令、工具、知识、记忆、版本、容量	规划、检索、调用工具并返回结果	追踪、评估门禁、发布与回滚

```
概念扩展：

机器学习 PaaS = 受治理的数据 + 作业 + 模型 + 端点 + MLOps

AI PaaS = 机器学习 PaaS 基础
          + 受治理的模型服务
          + 知识、提示词、工具、评估、安全与智能体
```

16.6 MLOps、LLMOps 与 AgentOps 是相互关联的运营回路

MLOps、LLMOps 与 AgentOps 常被描述为前后相继的替代关系，这种框架会产生误导。它们治理的是不同但相互重叠的资产集合与故障模式，一个企业应用可能三者都需要。

当组织训练、微调、注册、部署、监控或再训练模型时，MLOps 依然不可或缺。它强调可复现的数据与训练、模型血缘、部署晋级、漂移和再训练。LLMOps 强调预训练模型使用的生命周期：模型选择、提示词与上下文设计、检索、评估、安全、推理可观测性，以及令牌或吞吐量经济性。AgentOps 又加入轨迹、工具调用、记忆、人工批准、操作验证和结果级评估。

这些回路共享平台能力：都需要持久身份、版本化资产、环境与依赖控制、发布门禁、可观测性、策略、成本归因、事件响应和退役。若组织构建三套割裂的运营栈，就会重复采集证据，并在边界处形成盲区。AI PaaS 的目的不是重命名每种运维，而是通过共同的资源与证据契约连接这些回路。

数据、模型与智能体的运维交集

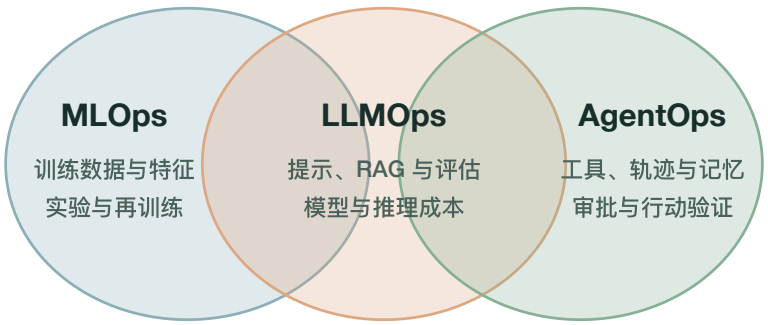


图 16-4 MLOps、LLMOps 与 AgentOps 通过共享的控制平面能力相互重叠。

运营回路	主要资产	典型生产问题	代表性发布门禁
MLOps	数据集、特征、代码、环境、作业、模型、端点	这个模型版本能否复现，并安全替换当前部署？	质量、公平性、稳健性、延迟与漂移标准
LLMOps	模型部署、提示词、知识、评估集、安全策略	在模型和上下文发生变化时，这个应用是否仍然有用、有依据、安全且具成本效益？	任务质量、依据性、安全、延迟和成本阈值
AgentOps	智能体版本、工具、记忆、轨迹、批准、结果	能否信任这个智能体在授权范围内行动，并产生可验证的业务结果？	轨迹、工具作用域、批准、结果与回滚标准

16.7 Azure 参考路径：Databricks、Machine Learning 与 Foundry

Azure 很适合作为教学参照，因为其平台边界让架构演进清晰可见。Azure Databricks 提供以湖仓为核心的统一数据、分析与 AI 环境；Azure Machine Learning 提供专用的机器学习项目与 MLOps 控制平面；Microsoft Foundry 则为模型服务、工具、知识、评估和智能体提供整合的资源模型。三种产品在某些开发与模型能力上存在重叠，但不应被视为前代依次被后代淘汰的三代产品。

典型的数据密集型 AI 系统可能使用 Azure Databricks 摄取和转换大规模数据集、管理湖仓表、计算特征并支持分析；使用 Azure Machine Learning 管理专用训练作业、可复用环境、模型资产、注册表、托管在线或批处理端点及机器学习监控；使用 Microsoft Foundry 在统一 Azure 控制平面下公开模型部署、项目级应用资产、评估、工具、知识连接与智能体服务。[3][4][5]

设计选择应服从生命周期。如果主要挑战是数据工程与湖仓治理，就从数据平台开始；如果系统训练或运营自定义机器学习模型，就使用机器学习生命周期平台；如果应用主要组合托管基础模型、检索、工具与智能体，就使用 AI PaaS 资源。许多生产架构会同时使用不止一种。目标不是减少产品名称数量，而是保持所有权和证据路径连贯。

Microsoft Foundry 2026 年的资源模型与本书论点尤其相关。Microsoft 文档描述了一个 Foundry 资源及其项目，并为模型、智能体与工具提供统一管理平面，其中包括 RBAC、网络、策略、追踪、监控和评估。[5] 这整合了早期 Azure AI 模式，但架构师仍应区分 Foundry 应用资源、Azure Machine Learning 资产以及底层数据资产范围。

Azure 层	最适合作为起点的场景……	关键托管对象	应保留的边界
Azure Databricks	系统以数据摄取、湖仓处理、分析及数据 / AI 协作为主	目录、表、笔记本、作业、集群或无服务器计算、MLflow 资产	不要假定数据工作区就是完整的应用或智能体控制平面

Azure 层	最适合作为起点的场景……	关键托管对象	应保留的边界
Azure Machine Learning	组织训练、微调、注册、部署或监控自定义模型	工作区、数据与环境资产、作业、流水线、模型、端点、监控器	不要把 MLOps 简化成模型目录或提示词工作流
Microsoft Foundry	应用组合托管模型、知识、工具、评估和智能体	Foundry 资源、项目、部署、评估、工具、知识与智能体版本	不要让 AI 应用资源脱离受治理的数据和生产所有权

16.8 阿里云路径：DataWorks、PAI 与 Model Studio

阿里云提供了类似的分层路径。DataWorks 是综合性数据平台，可与大数据和 AI 引擎集成，为仓库、数据湖和湖仓提供数据集成、开发、治理与分析。[11] PAI 是端到端 AI 开发平台，覆盖数据标注、开发、训练、部署和 AI 运维。其核心模块包括用于交互式开发的 DSW、用于分布式训练的 DLC、用于可视化工作流的 Designer、用于在线推理的 EAS，以及面向模型工作流的 Model Gallery。[12]

PAI 还说明机器学习 PaaS 控制平面如何向基础设施延伸。阿里云文档说明了为 DSW、DLC 和 EAS 提供算力的资源池、配额、工作区绑定、通用资源和高性能灵骏资源。[14] 关键抽象是分配契约：容量被购买或预配，通过配额切分，绑定到工作区，再由生命周期对象使用。相比让各团队自行获取任意 GPU 实例，这种方式更便于治理。

Model Studio 把重点转向托管模型服务和生成式 AI 应用。阿里云将其描述为一站式模型服务平台，通过官方 API 和 OpenAI 兼容 API 提供通义千问及第三方模型，支持多模态和按需调用。[15] 它还提供模型微调与面向应用的能力。[16] 从架构角度看，Model Studio 在 DataWorks 和 PAI 基础之上加入 MaaS 与 AI 应用资源；它并未让数据开发、分布式训练或模型运维变得多余。

产品边界和应用开发功能的区域可用性可能变化。为了形成持久架构，应记录工作负载使用的确切资源类型、区域、生命周期 API、身份模型、证据与生产状态。“门户里可用”并不是充分的控制平面契约。

16.9 腾讯云路径：TI-ONE 与 ADP

腾讯云 TI-ONE 为 AI 工程师提供一站式训练与推理平台。其平台文档涵盖开发机、训练任务、模型管理、推理服务、内置框架，以及计算资源的购买或分配。[17][18] 这种资源意图显然属于机器学习 PaaS：为开发、训练、注册、优化和提供模型服务提供托管环境。

腾讯云 ADP 运行在 AI 应用与智能体层。2026 年产品概览将其描述为企业 AI 应用平台，支持面向业务的智能体和多种构建模式，包括标准智能体、工作流、多智能体配置，以及可运行代码并调用技能、连接器与工具的独立工作区模式。它还支持 API、SDK 和企业渠道集成。[19]

因此，架构交接并不是“TI-ONE 被 ADP 取代”。TI-ONE 治理模型开发与服务生命周期，ADP 治理应用与智能体组合。数据服务和企业系统仍是上游与下游依赖项。若某工作负载要训练领域模型、构建受治理知识库并发布多智能体工作流，就可能横跨所有三个层。

16.10 火山引擎路径：VeDI、Ark、AgentKit 与 DataAgent

火山引擎把数据智能、模型服务与智能体运行时之间的分离表现得尤其清楚。VeDI 提供更广泛的数据智能环境及相关数据产品；DataAgent 被定位为企业 AI 数据专家，利用组织的数据资产进行分析、洞察和行动。[24] 它应被视为数据智能解决方案层，而不是对所有数据工程或 AI 平台资源的通用替代品。

Ark 提供模型服务层。火山引擎文档说明 Ark 平台下的推理、评估、微调及相关模型生命周期服务。[20][21] AgentKit 又加入云托管智能体运行时、基于 CLI 的生命周期操作、工具、评估和可观测性。其 CLI 支持初始化、配置、构建、部署与运维，而运行时则是智能体的托管执行与生命周期单元。[22][23]

由此形成的路径与本章论点一致：VeDI 及相关数据服务治理数据与业务语义；Ark 治理模型服务与优化；AgentKit 治理智能体执行与生命周期；DataAgent 把选定能力封装成企业数据智能体验。这些层可以集成，但不必假装它们属于同一种资源类型。



图 16-5 Azure、阿里云、腾讯云与火山引擎围绕相近的数据、机器学习、模型服务和智能体层，划定了不同产品边界。

工程意图	Azure	阿里云	腾讯云	火山引擎
数据开发与治理	Azure Databricks 加 Azure 数据与治理服务	DataWorks 及其集成的数据引擎	腾讯云数据与分析服务	VeDI 及相关数据产品
交互式 AI 开发	Azure ML 笔记本 / Databricks 笔记本 / Foundry 项目工具	PAI DSW / Designer	TI-ONE 开发机与工作台	Ark 开发能力与平台工具
训练与优化	Azure Machine Learning 作业与计算	PAI DLC、Designer、Model Gallery	TI-ONE 训练与优化	Ark 微调与模型服务
模型部署与推理	Azure ML 托管端点 / Foundry 模型部署	PAI EAS / Model Studio 模型 API	TI-ONE 推理与托管模型服务	Ark 在线推理与模型单元
知识与生成式应用	Foundry 知识、工具与项目资源	Model Studio 知识与应用能力	ADP 知识与应用模式	Ark 与 AgentKit 的知识 / 工具集成
智能体运行时与生命周期	Foundry Agent Service	Model Studio / PAI 智能体应用能力	ADP	AgentKit
提供商特定注意事项	不要把 Databricks、Azure ML 与 Foundry 压缩成一个工作区契约	不要推断 PAI 与 Model Studio 的功能或区域对等性	不要把 ADP 视为模型生命周期的替代品	不要把 DataAgent、Ark 与 AgentKit 视作一种通用资源

16.11 架构工作表：分离资源家族

本章不包含完整部署实验，其实践练习是一份架构工作表。目标是在选择提供商或产品之前，避免“AI 平台”一词掩盖彼此不兼容的生命周期与责任边界。

选择一个用例，例如服务运营知识助手、制造质量分析器、金融风险解释服务或云成本建议助手。用一句话描述业务结果，然后分别盘点各个资源家族，不要从产品名称开始。



图 16-6 工作表分离数据、模型、知识、工具和智能体资源，再通过共享治理记录将其重新连接。

工作表字段	要回答的问题	最低制品要求
业务结果	谁将获得价值，哪项决策或任务会改进，如何衡量成功？	一条结果陈述和一个单位价值指标
数据资源	需要哪些来源、转换、所有者、分类、新鲜度与血缘？	来源清单和数据流草图
模型资源	模型是直接使用、微调还是训练？涉及哪些部署、区域、容量与退役风险？	模型决策记录
知识资源	哪些内容会变得可检索？如何准入、刷新、授权、评估和引用？	知识生命周期与资格规则
工具资源	哪些系统可以读取或更改？通过哪个身份？具有什么风险与批准要求？	工具能力卡与授权边界
智能体或应用资源	存在哪些定义、工作流、运行时、记忆、会话与发布对象？	智能体 / 应用对象模型与生命周期
评估	哪些数据集、指标、评分器和阈值用于把守发布？	评估套件与发布门禁
共享治理	每项资源由谁负责？如何联网、观测、计费、保留和退役？	责任矩阵与证据计划
提供商映射	哪种产品提供每项资源？哪些地方不存在运营等价性？	带不等价说明的提供商映射

1id1.net

验收规则

只有每个资源家族都具备所有者、生命周期、身份路径、证据源、成本边界和退役操作，工作表才算完整。产品徽标不是架构决策。

16.12 设计规则与反模式

AI PaaS 采用失败，很少是因为模型端点无法调用，更多是因为外围资源图未经设计。以下反模式尤为常见。

反模式	失败原因	首选设计规则
模型 API 单一文化	每项需求都被简化为选择并调用模型	模型访问只是更大规模受治理应用生命周期中的一项数据平面能力
单体式“AI 平台”框	数据、模型、知识、工具和智能体所有权变得不可见	分离资源家族，再通过显式契约重新连接
把训练流水线当作完整应用	提示词、检索、工具与用户工作流被排除在发布证据之外	把 LLMOps 与 AgentOps 资源加入发布图
没有源代码控制的门户驱动资产	实际生产状态无法审查或复现	导出或定义持久配置，对其进行版本控制，并记录晋级证据
重复数据平面	各团队用不一致的语义，把数据复制进私有索引与笔记本	治理共享来源与转换，同时保留工作负载隔离
把 RAG 等同于向量数据库	遗漏授权、新鲜度、来源、评估与回滚	把知识视为托管资源生命周期
把智能体等同于提示词加工具	遗漏身份、工具权限、追踪、评估、记忆与退役	把智能体视为受治理云资源
没有退役路径	旧模型、索引、端点、提示词与智能体仍可调用并持续计费	在生产之前定义弃用、迁移、保留、删除与清除
产品名称等价	相似标签掩盖不同的资源、API、证据与运营契约	转换工程意图，并记录不等价性

16.13 CloudDesk 复用：平台清单与生命周期缺口评估

CloudDesk FinOps Agent 以后可使用本章资源模型，解释 AI 工作负载为何产生相应成本以及治理缺口所在。本章不实现该智能体，而是定义可供后续工具层安全复用的只读能力。

平台清单工具应按家族对资源分类，而不是只列出提供商对象类型。血缘工具应把数据转换连接到模型、知识索引、应用与智能体。生命周期缺口评估器应检测缺少所有者、版本、评估证据、监控、成本归因或退役计划的资源。这些能力对 FinOps 很有用，因为孤

立或重复的 AI 资产既造成经济浪费，也带来治理风险。

```
capability: ai.platform.architecture.map
risk_level: R0_read_only
inputs:
  scope: cloud account, subscription, project, or workspace
  resource_families: [data, ml, model, knowledge, tool, evaluation, agent]
outputs:
  resource_graph: []
  ownership_gaps: []
  lifecycle_gaps: []
  duplicated_capabilities: []
  evidence_sources: []
constraints:
  - never infer equivalence from product names alone
  - preserve provider resource IDs and regions
  - distinguish missing evidence from evidence of absence
```

能力	用途	最低权限
data.lineage.inspect	解释数据、特征、索引与模型资产的上游和下游依赖关系	读取目录、流水线、作业和血缘元数据
ml.asset.inventory	盘点工作区、作业、模型、端点和监控对象	读取机器学习 PaaS 控制平面元数据
ai.platform.architecture.map	将提供商资源归入共同资源家族，并记录不等价性	读取模型、知识、评估、工具和智能体元数据
ai.lifecycle.gap.assess	识别缺少所有权、评估、证据、成本归因或退役计划的资源	对配置、活动、可观测性和账单元数据的只读访问

16.14 要点

- 数据服务是 AI 工程的基础，因为它们为训练和检索共同治理质量、血缘、授权、新鲜度与保留。
- 机器学习 PaaS 通过工作区、资产、作业、流水线、注册表、端点、监控和 MLOps 证据，将机器学习生命周期产品化。
- AI PaaS 是控制平面的扩展，而非替代品：它加入托管模型服务、提示词、知识、工具、评估、安全策略和智能体。
- MLOps、LLMOps 与 AgentOps 治理相互重叠的资产集合，应共享身份、版本控制、可观测性、策略、成本、发布与退役契约。
- Azure Databricks、Azure Machine Learning 与 Microsoft Foundry 展示了彼此不同但相互连接的数据、机器学习和 AI 应用生命周期。

- 阿里云、腾讯云与火山引擎通过不同产品边界公开相近的工程层；必须记录不等价性。
- 稳健的架构应先分离数据、模型、知识、工具、评估和智能体资源，再选择提供商产品。
- 成熟的 AI 工程单元不再是模型文件或 API 调用，而是具备所有者、证据、经济性和退役路径的受治理资源图。

参考文献

[1] 美国国家标准与技术研究院：《人工智能风险管理框架（AI RMF 1.0）》，NIST AI 100-1，2023 年。https://doi.org/10.6028/NIST.AI.100-1

[2] 美国国家标准与技术研究院：《人工智能风险管理框架：生成式人工智能概况》，NIST AI 600-1，2024 年。https://doi.org/10.6028/NIST.AI.600-1

[3] Microsoft：《什么是 Azure Databricks？》，Microsoft Learn，更新于 2026 年 7 月 28 日。https://learn.microsoft.com/en-us/azure/databricks/introduction/

[4] Microsoft：《什么是 Azure Machine Learning？》，Microsoft Learn，更新于 2025 年 9 月 9 日。https://learn.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-machine-learning

[5] Microsoft：《什么是 Microsoft Foundry？》，Microsoft Learn，更新于 2026 年 7 月 20 日。https://learn.microsoft.com/en-us/azure/foundry/what-is-foundry

[6] Microsoft：《Microsoft Foundry Models 概览》，Microsoft Learn，更新于 2026 年 7 月 28 日。https://learn.microsoft.com/en-us/azure/foundry/concepts/foundry-models-overview

[7] Microsoft：《Azure Machine Learning 中的 MLOps》，Microsoft Learn。https://learn.microsoft.com/en-us/azure/machine-learning/concept-model-management-and-deployment

[8] MLflow 项目：MLflow Tracking 与 Model Registry 文档。https://mlflow.org/docs/latest/

[9] OpenLineage 项目：OpenLineage 文档。https://openlineage.io/docs/

[10] Microsoft：《Microsoft Purview 中的数据血缘》，Microsoft Learn。https://learn.microsoft.com/en-us/purview/concept-data-lineage

[11] 阿里云：《什么是 DataWorks？》，更新于 2026 年 1 月 12 日。https://www.alibabacloud.com/help/en/dataworks/user-guide/what-is-dataworks

[12] 阿里云：《什么是机器学习平台 PAI？》，更新于 2026 年 6 月 23 日。
https://www.alibabacloud.com/help/en/pai/product-overview/what-is-machine-learning-platform-for-ai/

[13] 阿里云：PAI 产品架构，更新于 2026 年 6 月 11 日。
https://www.alibabacloud.com/help/en/pai/product-overview/service-architecture

[14] 阿里云：AI 计算资源概览，更新于 2026 年 7 月 9 日。
https://www.alibabacloud.com/help/en/pai/overview-of-ai-computing-resources

[15] 阿里云：《什么是阿里云 Model Studio？》，更新于 2026 年 7 月 10 日。
https://www.alibabacloud.com/help/en/model-studio/what-is-model-studio

[16] 阿里云：模型微调简介，更新于 2026 年 7 月 14 日。
https://www.alibabacloud.com/help/en/model-studio/model-training-overview

[17] 腾讯云：TI-ONE 平台概览，更新于 2026 年。
https://www.tencentcloud.com/document/product/1141/45261

[18] 腾讯云：TI-ONE 模型管理简介，更新于 2026 年。
https://www.tencentcloud.com/document/product/1141/77102

[19] 腾讯云：智能体开发平台产品概览，更新于 2026 年 7 月 23 日。
https://www.tencentcloud.com/document/product/1254/69956

[20] 火山引擎：Ark 产品文档——推理、评估与微调服务。
https://docs.volcengine.com/docs/82379

[21] 火山引擎：在 Ark 中创建模型评估任务。
https://www.volcengine.com/docs/82379/1153149

[22] 火山引擎：AgentKit CLI 概览与生命周期管理，更新于 2026 年 6 月 23 日。
https://www.volcengine.com/docs/86681/2085680

[23] 火山引擎：AgentKit 数据可观测性，更新于 2026 年 8 月 10 日。
https://www.volcengine.com/docs/86681/1873528

[24] 火山引擎：DataAgent 产品文档。https://docs.volcengine.com/docs/86760?lang=zh

[25] Databricks：《什么是数据湖仓？》。
https://docs.databricks.com/en/lakehouse/index.html

[26] Databricks：Unity Catalog 与数据治理文档。https://docs.databricks.com/en/data-governance/unity-catalog/index.html

[27] Feast 项目：Feature Store 文档。https://docs.feast.dev/

[28] ISO/IEC 5338:2023：《信息技术——人工智能——AI 系统生命周期过程》。

[29] ISO/IEC 42001:2023：《信息技术——人工智能——管理体系》。

[30] CNCF：云原生人工智能白皮书与生态资源。https://www.cncf.io/

草稿范围说明

本章有意止于平台架构与资源家族。第 17 章深入治理模型资源；第 18 章治理知识；第 19 章治理智能体平台；第 20 章治理作为智能体工具的云能力。自定义无服务器智能体运行时和租户隔离属于另一本书《无服务器 AI 智能体开发》。

第五篇 / 第 17 章

从模型 API 到 AI PaaS 资源

把推理视为具有版本、身份、配额、网络、遥测、安全与成本语义的受治理资源

模型 API 证明软件能够发送提示词并接收响应。AI PaaS 资源则证明，同一能力可以被命名、版本化、授权、隔离、观测、计费、变更和退役，同时不丧失运营问责。

本章内容

- 为何一次成功的推理请求只是数据平面事件，而不是完整的生产架构
- 模型资源契约：模型身份、部署、端点、授权、网络、配额、安全、遥测、计费与生命周期
- 为何目录模型、模型版本、部署、端点、项目和 AI PaaS 资源是不同对象
- MaaS 如何提供托管推理，却不能单独提供完整的 AI PaaS 控制平面
- 共享在线、区域、数据区、预配、批处理、专用、微调和自定义部署如何表达不同运营契约
- 为何 API 密钥是便利的引导凭据，却不适合作为受治理生产工作负载的长期身份
- 区域、私有网络、数据处理作用域和出站依赖如何界定真正的推理边界
- 为何配额、速率限制、容量、背压、预算与服务目标不能混为同义词
- 模型版本、升级策略、金丝雀发布、回滚、弃用和退役如何形成托管生命周期
- 护栏与安全策略如何成为与部署和智能体关联的版本化资源
- 将推理作为服务而非不透明 API 调用来运营，需要哪些指标和证据
- 如何检查并调用托管模型部署，再组装可验证的模型资源证据包
- Azure、阿里云、腾讯云与火山引擎如何通过不同资源模型公开相近的工程意图

新应用的首次模型调用往往令人振奋。几行代码就产生流畅输出，开发团队由此证明创意在技术上可行。这个证明很重要，却只回答了一个狭窄问题：应用能否访问模型端点并收到响应？它没有回答由哪个模型版本处理请求、授权了哪个主体、请求可以在哪里处理、受哪个配额池约束、由哪项安全策略评估、如何测量延迟与令牌用量、成本是多少，以及模型版本退役时会发生什么。

这些尚未回答的问题并非次要管理事项，它们决定能否把生产角色托付给推理。AI PaaS 始于孤立 API 调用结束之处：它把模型访问转化为具有显式控制平面状态和可检查数据平面证据的可治理云资源。该资源仍可能公开 OpenAI 兼容 API 或提供商特定 API，但 API 现在只是更大托管契约的一个接口。



图 17-1 模型 API 证明可调用性；模型资源提供外围生产契约。

17.1 模型 API 是数据平面契约，而不是生命周期

推理 API 定义调用方如何提交输入并接收输出。它规定一种协议：端点、身份验证机制、请求字段、流式传输行为、工具调用结构和响应对象。这是数据平面契约；请求是既有服务执行的一项操作。

控制平面回答的是另一类问题。它创建允许推理发生的服务边界，记录部署名称、模型版本、容量模式、区域、网络访问、角色分配、安全配置、诊断路由、配额分配和删除状态，还让基础设施即代码、策略、清单、审计和成本系统可以看见这些决策。

表 17-1 数据平面问题与控制平面问题

数据平面：执行推理	控制平面：治理推理
哪个端点接受请求？	哪个 AI 资源和部署拥有该端点？
API 需要什么请求架构？	批准了哪个模型名称、版本、格式和部署类型？
如何验证请求身份？	哪些主体可以部署、调用、评估、监控或退役该资源？
返回什么响应、数据流或工具调用？	哪些护栏、遥测、配额和成本策略约束该操作？
此次请求成功了吗？	组织能否解释变更历史、失败、限流、成本与退役？

一种常见架构错误是把 API 密钥和基址 URL 当成服务。它们只是凭据和寻址信息，并非服务边界。如果团队无法独立于应用代码盘点部署，就无法可靠治理该部署；如果部署可以在不留下运营证据的情况下更改版本，应用就无法稳定解释行为变化；如果所有调用方

共享同一把不受限制的密钥，授权就退化为“持有秘密”。

注意：本章使用“模型资源”作为与提供商无关的工程术语。云平台使用不同名称，例如模型部署、推理服务、端点、在线服务、模型单元或模型端点。关键不在产品标签，而在于该对象是否具有明确的生命周期、身份、网络、配额、遥测、安全和计费语义。

17.2 模型资源契约

生产部署应当无需阅读应用源代码就能描述。这种描述就是模型资源契约。它不一定对应单一物理资源，可能由 AI 账户、项目、部署子资源、端点、角色分配、私有端点、诊断设置、策略绑定和账单记录共同表示。这些对象合在一起，构成应用使用的受治理单元。

表 17-2 模型资源契约的最低字段

契约字段	运营问题	证据
资源身份	哪个云资源拥有该部署？	提供商资源 ID、账户、项目、区域
模型身份	正在使用哪个模型，以及哪个不可变或由提供商定义的版本？	模型名称、格式、版本或快照
部署身份	调用方使用哪个稳定名称？	部署或端点名称与资源 ID
容量契约	如何提供并限制容量？	部署类型、SKU、容量单位、并发或吞吐量设置
授权	谁可以部署、调用、检查或删除？	角色分配、策略、工作负载身份
网络边界	流量可以从哪里进入和离开？	公共访问状态、私有端点、DNS 与出站规则
处理边界	提供商可以在哪里处理请求？	区域、数据区、全局作用域、提供商记录的行为
安全策略	哪些控制措施评估输入、输出和工具？	护栏或负责任 AI 策略的名称与版本
遥测契约	采集并保留哪些信号？	指标、日志、追踪、请求 ID、诊断目的地
计费契约	哪些单位产生费用？	输入 / 输出令牌、容量单位、小时、作业、数据移动
升级策略	底层模型可以如何变化？	固定版本、自动升级选项、金丝雀计划
退役契约	如何发现并完成弃用？	提供商通知、清单查询、迁移证据、删除记录

该契约建立了关键分离：应用请求一个部署名称，运维人员则保留检查该名称含义的能力。稳定部署名称可避免每个调用方硬编码易变的模型标识符。然而，调用方边界的稳定性不能抹去审计、遥测与发布记录中的版本身份。

该契约也让平台自动化成为可能：策略可以拒绝公共访问；清单可以发现使用过时模型版本的部署；FinOps 工具可以把令牌消耗关联到项目和业务所有者；可观测性可以区分由版本变更导致的延迟回归和由应用发布导致的回归；智能体工具则可以限定为只读检查或已批准的部署操作。

17.3 模型、版本、部署、端点、项目与资源

AI 平台经常把多个名称紧挨着展示，促使团队把它们压缩成一个心理对象。它们应当保持区分。

表 17-3 托管推理资源图中的对象

对象	含义	典型变更频率
目录模型	提供商发布、可供选择的一项能力或模型家族。	由提供商控制；目录频繁变化
模型版本	模型的一次特定提供商发布、快照或修订。	提供商发布或退役版本时变化
部署	由客户治理的绑定，组合模型版本、容量模式、策略与稳定部署名称。	通过受控发布操作变更
端点	用于推理的数据平面地址与协议。	通常稳定；其后路由可能变化
项目或工作区	应用、智能体、评估、连接和制品的协作与开发边界。	随团队和解决方案组织方式变化
AI PaaS 资源或账户	安全、网络、计费、配额与模型部署的顶层云边界。	长期存在；通过云资源管理进行治理

项目尤其容易被误解。项目可以隔离开发资产、连接、评估运行和团队权限，但不会自动为每个项目创建物理专用模型。反过来，即使项目制品彼此分离，一个部署也可以由多个应用共享。必须检查资源层级，而不能从门户导航推断。



图 17-2 与提供商无关的模型资源图，以 Azure Foundry 层级作为参考形态。

17.4 MaaS 是一个层，而不是完整 AI PaaS

模型即服务（MaaS）免除了训练、托管和缩放基础模型的负担。调用方把请求发送到托管端点，通常按令牌、请求、图像、音频时长或其他消费单位付费。MaaS 是重要的平台进步，因为它把稀缺模型基础设施转化为按需能力。

然而，MaaS 本身没有定义完整的应用工程环境。AI PaaS 加入构建和运营 AI 原生应用所需的外围系统：项目、模型目录与部署、身份、私有网络、知识、评估、提示词与智能体资产、工具连接、安全、遥测、策略、成本和生命周期自动化。提供商可能用一个工作室或平台名称营销这些能力，但工程区别仍然有用。

表 17-4 MaaS 与 AI PaaS 相关但不相同

能力	MaaS 的贡献	AI PaaS 的扩展
推理	托管模型执行与缩放	部署生命周期、路由、策略、评估与运营证据
身份	API 密钥或服务身份验证	云 IAM、工作负载身份、限定作用域的角色、职责分离
数据与知识	请求负载和可选文件接口	受治理的数据连接、知识资源、摄取与检索
应用工程	SDK 与 API 兼容性	项目、智能体、工具、工作流、追踪、发布与治理

能力	MaaS 的贡献	AI PaaS 的扩展
运维	提供商服务可用性	客户 SLO、配额、诊断、成本分配、变更与退役

这种区分可避免错误的架构结论：应用能通过兼容 API 调用模型，就意味着组织已经拥有 AI 平台。API 兼容性只是降低集成阻力，并不会自动建立所有权、隔离、质量、变更控制或成本问责。

17.5 部署家族就是运营契约

同一模型家族可能通过多种部署类型提供。它们不只是采购选项，还会改变请求可以在哪里处理、如何提供容量、负载下延迟如何表现、如何计算成本，以及哪些运营责任仍由客户承担。

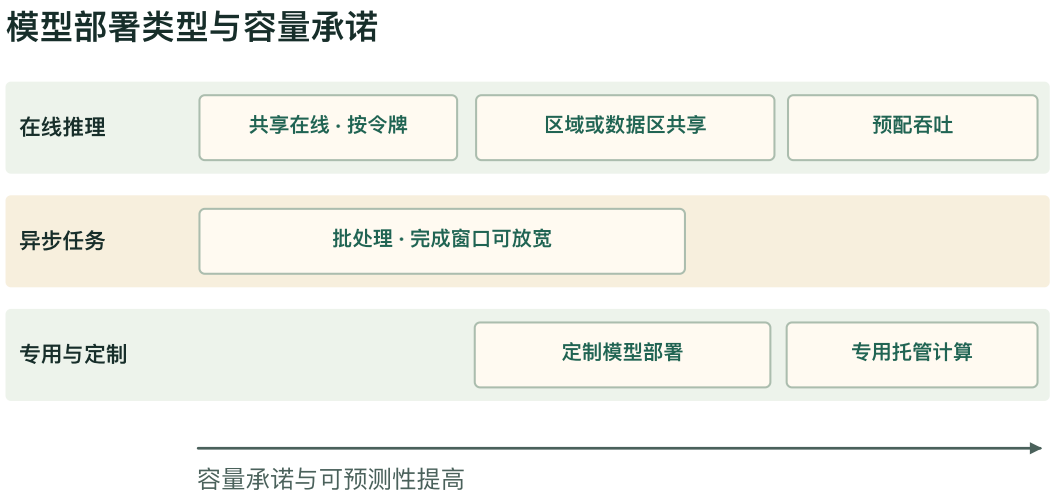


图 17-3 部署家族的概念权衡空间。产品名称会变化，容量与延迟的权衡则长期存在。

表 17-5 常见部署家族

部署家族	最佳适用场景	主要工程权衡
共享在线、按用量计费	交互式原型和可变的生产需求	采用迅速且有弹性；共享容量下的延迟和吞吐量较难预测
区域或数据区共享在线	有明确处理边界要求的工作负载	处理作用域更窄，但可能减少模型或容量选择

部署家族	最佳适用场景	主要工程权衡
预配吞吐量或预留容量	具有严格吞吐量目标的稳定高容量工作负载	容量承诺提升可预测性，却也产生利用率风险
批处理或异步推理	离线增强、评估、迁移和大型非交互式作业	以延迟完成换取较低紧迫度，通常还有更好的单位经济性
专用托管计算	专用模型、专用加速器、自定义容器或更强运行时控制	需要更明确的计算规模、热容量、修补边界与小时成本
微调或自定义模型部署	仅靠提示词和检索无法实现的领域行为	增加训练数据治理、版本血缘、评估与模型生命周期

有用的决策从工作负载形态出发，而不是模型热度。交互式聊天需要首令牌时间和尾延迟目标；批量文档分类需要完成时间和作业故障恢复；智能体工具选择需要低延迟和可预测的速率限制行为；评估工作负载需要可复现性和成本控制。单一部署类型很少能同时优化这四类场景。

处理作用域也是契约的一部分。描述为全局的部署可能允许提供商在更广地理范围内路由推理，同时仍遵守该服务定义的数据处理承诺。区域或数据区部署会收窄允许的处理边界，但容量或模型可用性可能不同。确切语义因提供商而异，必须以当前官方文档核验。

17.6 身份：从密钥到工作负载主体

API 密钥之所以有吸引力，是因为它把身份验证压缩成一个字符串。它适合初期探索，但便利性掩盖了多项治理弱点。密钥通常代表资源，而不是工作负载；难以区分由哪个应用发起调用，难以使用细粒度角色约束，也很容易被复制进本地配置、笔记本、CI 日志或桌面工具。

生产设计应优先使用能够获取短期令牌、并通过云身份系统授权的工作负载身份。在 Azure 中，Microsoft Entra ID 与 Azure RBAC 允许不同主体分别管理 Foundry 资源、创建部署、调用推理、检查监控或使用智能体。阿里云 RAM、腾讯云 CAM 与火山引擎 IAM 体现了同一总体意图，尽管角色名称和资源作用域有所不同。

表 17-6 按责任分离身份

主体	所需能力	通常应当缺少的能力
平台部署者	创建或更新 AI 资源、项目、部署、策略和诊断	日常应用推理和业务数据访问
推理工作负载	调用已批准部署，且仅读取所需运行时元数据	创建部署、分配角色、更改网络、删除资源

主体	所需能力	通常应当缺少的能力
评估者	调用已批准部署、写入评估制品、读取质量遥测	生产修改工具和不受限制的秘密
运维读取者	读取部署状态、指标、配额、成本、运行状况和退役通知	推理内容，除非事件策略明确允许
紧急管理员	具有强审计的限时紧急控制平面权限	长期日常访问

身份还区分管理平面访问与数据平面访问。能够在云门户中查看资源的主体，仍可能无法调用模型；相反，持有宽泛 API 密钥的调用方可能在没有狭窄云角色的情况下调用模型。这种不对称必须经过测试，不能凭假设判断。

注意：秘密管理器改善密钥的存储与轮换，却不会把密钥变成工作负载身份。更强的模式是在平台支持基于令牌的工作负载身份验证时消除长期秘密，只对尚不能使用联合身份或托管身份的依赖项使用秘密存储。

17.7 网络与处理边界

托管模型可能在外部托管，但仍可参与私有应用架构。关键区别在于入站可达性、名称解析、出站依赖与提供商处理作用域。

表 17-7 必须记录的四种边界

边界	控制问题	典型机制
入站网络	该端点能否从公共互联网访问？	公共网络标志、防火墙规则、私有端点、专用 DNS
调用方出站	工作负载可以从哪些子网、服务或网关发送请求？	网络安全策略、出站防火墙、服务端点、代理
提供商处理	提供商可以在哪个地理范围处理推理请求？	区域、数据区或全局部署契约
已连接服务	AI 平台可以调用哪些存储、搜索、工具、遥测或安全服务？	托管私有端点、服务身份、允许列表、连接资源

私有连接是会产生数据平面后果的控制平面决策。在 Azure 中，AI 资源可以禁用公共网络访问，并向虚拟网络公开私有端点。专用 DNS 让普通服务主机名从网络内部解析为私有地址。因此，应用应保留一个逻辑端点，由网络策略决定允许的路由。

私有入口不会自动约束每条出站路径。如果智能体可以调用任意工具、项目连接到外部存储，或评估作业需要软件包存储库，这些路径就需要单独设计。网络隔离不是模型端点上的一个复选框，而是完整 AI PaaS 解决方案的图属性。

17.8 配额、速率限制、容量、背压与预算

托管推理引入了多种经常被混淆的数值控制。它们用途不同：配额授予使用容量的资格；速率限制保护共享服务运行；部署容量分配或预留一部分服务能力；背压在请求超过可用吞吐量时保护应用；预算报告财务阈值；SLO 定义可接受的服务行为。

表 17-8 外表相似、含义不同的控制

控制	回答的问题	不保证什么
配额	该订阅、账户、项目或区域可为某个模型或部署类别分配多少？	实际吞吐量、低延迟或低成本
速率限制	请求或令牌可以多快到达而不触发限流？	预留容量或业务优先级调度
预配容量	承诺或预留了多少吞吐量？	高效利用或正确的工作负载设计
背压	可用容量耗尽时，调用方如何排队、丢弃、重试或降级？	额外提供商容量
预算	累计成本应在何时触发通知或工作流？	自动阻止推理或硬性执行支出上限
SLO	什么样的延迟、可用性、错误或质量行为可以接受？	它本身并不是创建容量的机制

配额常在提供商定义的作用域内分配，例如订阅、账户、区域、模型家族、版本或部署类型。该作用域会随平台演进而变化。团队若只存储一个数值限制，就会丢失解释它所需的上下文。证据应记录配额作用域、单位、测量窗口、当前分配、部署容量和来源时间戳。

应用必须假设限流可能发生。稳健的客户端会使用带抖动的有界重试，遵循重试指引，限制并发工作，并把非交互式工作负载路由到队列或批处理服务。智能体运行时不应让每次工具调用在没有共享请求预算的情况下各自重试；这种行为可能把容量事件放大成成本事件。

17.9 模型版本、升级与退役

模型名称不是充分的发布标识符。提供商会修订模型家族、发布快照、更改默认版本、把部署迁移到较新版本并退役旧模型。即使 API 兼容性不变，响应风格、推理行为、安全敏感度、令牌用量、延迟和工具调用结构也可能变化。

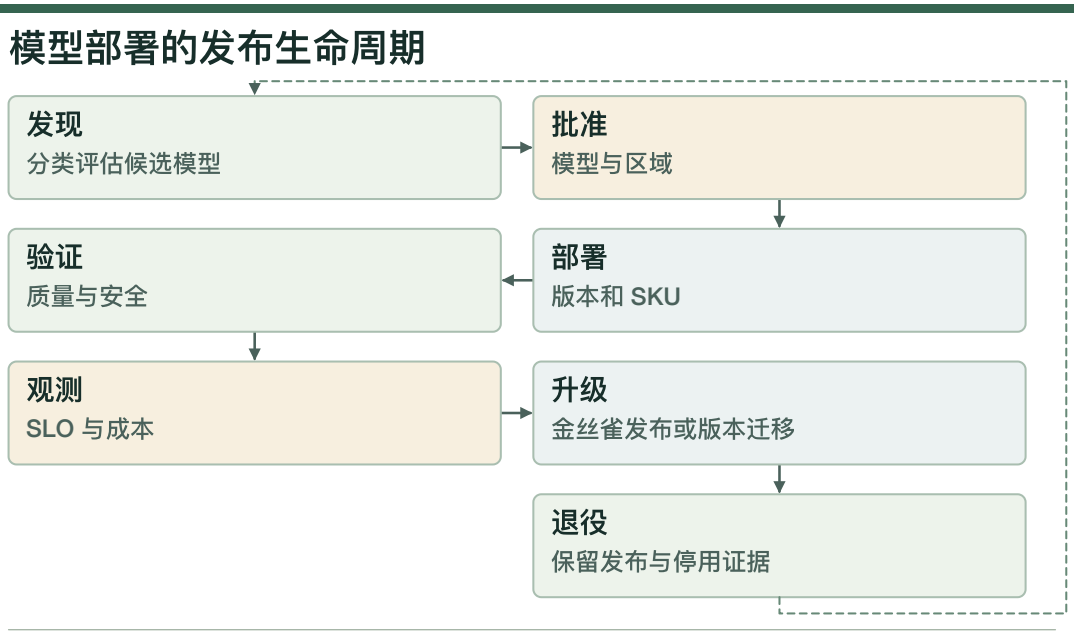


图 17-4 部署生命周期把模型变更转化为显式发布过程。

表 17-9 模型升级所需证据

阶段	所需证据
发现	提供商通知、候选模型 / 版本、按区域和部署类型划分的可用性
批准	用例适配度、数据边界、许可证或提供商条款、风险分类、所有者
部署候选版本	独立部署名称或受控升级选项、容量与策略绑定
验证	黄金集质量、安全、工具行为、延迟、令牌与成本比较
金丝雀发布	流量比例或选定租户、回滚阈值、可观测性维度
晋级	关联测试与生产证据的决策记录
退役	调用方迁移、依赖清单、最终成本与审计导出、删除确认

稳定部署名称是一种抽象，并不是静默修改的许可。如果平台支持自动版本升级，组织应决定这种行为在哪些场景可接受。低风险实验可以接受自动迁移；受监管或使用工具的生产智能体可能需要固定版本和明确迁移窗口。无论哪种方式，当前实际版本都必须可观测。

退役必须作为架构的一部分。部署清单应回答哪些应用和智能体仍在使用即将退役的版本；兼容性测试应在截止日期前运行；替代版本应在删除旧部署之前完成部署与评估；证据在删除后还应保留足够长时间，以支持审计和事件复盘。

17.10 把安全策略与护栏作为资源

安全行为不应是不可见的提供商默认值。生产系统需要一项能够命名、版本化、审查、分配、测试并独立于应用代码变更的明确策略。例如，Microsoft Foundry 护栏把多种控制组织成具名集合，可以评估模型输入和输出，并在受支持的智能体场景中评估工具调用与工具响应。

表 17-10 护栏控制家族

控制家族	用途	工程证据
内容类别	依据策略阈值检测或阻止有害类别	策略名称、阈值、阻止结果、误报审查
提示词攻击检测	识别试图覆盖系统意图或操纵指令的行为	检测结果、来源上下文、响应操作
间接攻击检测	检查检索内容或工具提供内容中的恶意指令	文档 / 工具来源、控制结果、隔离操作
PII 或敏感数据控制	检测、脱敏、阻止或路由敏感内容	数据类别、策略决策、保留规则
任务遵循 / 工具安全	检查计划的或返回的工具行为是否仍在任务和策略范围内	工具名称、请求的操作、策略判定、批准
自定义业务规则	应用领域特定分类器、允许列表或后置条件	规则版本、所有者、测试语料库、例外流程

护栏不能替代应用授权。内容筛选器无法决定主体是否有权关闭虚拟机；提示词攻击检测器不能取代工具允许列表。安全控制应与身份、数据授权、网络边界、架构验证、批准和审计共同运行。

护栏变更应像模型变更一样接受测试。更严格的策略可以改善风险态势，却可能产生不可接受的拒绝率；更宽松的策略可能提高完成率，却暴露敏感内容。因此，部署记录应绑定产生每项评估结果的模型版本与护栏版本。

17.11 可观测性与推理 SLO

即使采用托管推理，若团队只收集应用异常，运营仍可能不透明。服务必须公开请求级或聚合证据模型，把工作负载身份、部署、模型版本、延迟、令牌用量、限流、安全结果和计费单位连接起来。

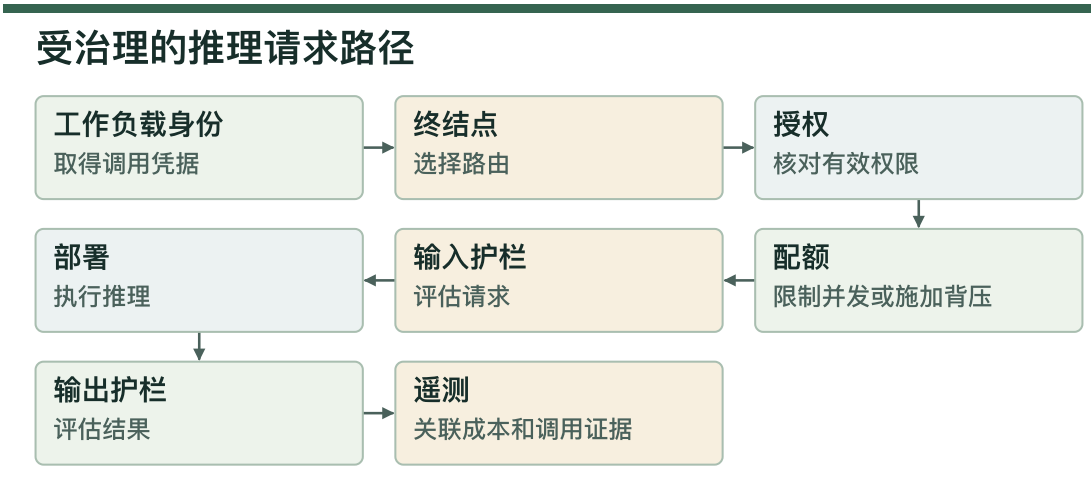


图 17-5 受治理请求路径同时产生控制平面状态与数据平面证据。

表 17-11 最低推理信号

信号	重要原因	有用维度
请求数与成功率	区分需求与可靠性，并发现静默放弃	部署、项目、工作负载、区域、响应类别
端到端延迟	显示用户可见延迟	部署、操作、流式 / 非流式、客户端区域
首令牌时间	区分模型启动 / 排队延迟与生成时长	部署类型、模型版本、请求类别
输入与输出令牌	解释工作负载形态与成本	模型、部署、租户 / 项目、功能
受限流的请求	显示容量压力和客户端重试需求	配额作用域、部署、调用方、分钟 / 小时窗口
4xx 与 5xx 响应	区分调用方、策略与服务故障	状态码、操作、部署、主体
安全阻止与转换	衡量护栏影响与潜在攻击	策略版本、类别、输入 / 输出 / 工具阶段
版本与配置变更事件	解释仅凭请求指标无法说明的行为变化	执行者、资源 ID、旧值 / 新值、相关 ID

表 17-12 推理 SLO 示例

目标	指标示例	支持的决策
交互响应性	99% 的成功流式请求在约定阈值内产生首个令牌	缩放、重路由、降级或暂停发布
请求可靠性	30 天内成功的非策略响应数 / 合格请求数	基于错误预算的发布决策
容量韧性	合格请求的限流比例低于阈值，且队列时长有界	增加容量或更改工作负载调度
安全可用性	已批准评估集上的策略误报率低于阈值	调优策略或路由到审查
成本效率	每个获接受业务结果的实际成本处于目标范围	更改模型、提示词、缓存、批处理或容量模式

不能仅因原始提示词和响应对调试有用，就默认记录它们。它们可能包含个人数据、商业秘密、凭据、检索文档或工具输出。应优先使用请求 ID、哈希、结构化属性、抽样脱敏内容和受控诊断采集，并明确规定保留和访问。

控制平面贡献了数据平面无法自行生成的变更事件。角色分配变更、部署更新、网络变更、护栏更新、配额变化和删除都应与请求遥测相关联。没有这种关联，操作人员虽能观察到回归，却无法确定发生了什么变化。

17.12 计费语义与单位经济性

模型部署具有计费契约，而不是统一价格。共享在线推理可能分别对输入和输出令牌计费；某些服务区分缓存输入、图像、音频或工具相关操作；预配吞吐量按承诺容量计费；专用计算按运行中的基础设施计费；批处理可能使用不同单价和完成模式；微调还会增加训练与托管模型成本。

表 17-13 托管推理中的成本组成

组成部分	典型计费单位	设计杠杆
输入处理	输入令牌、字符、图像、音频时长或请求	提示词大小、检索精度、压缩、缓存
输出生成	输出令牌、图像、音频或生成媒体	输出限制、响应格式、模型选择、后处理
预留容量	预配吞吐量单位或容量小时	承诺规模、调度、利用率
专用运行时	加速器 / CPU / 内存小时与附加存储	自动缩放、热池、模型大小、运行时效率

组成部分	典型计费单位	设计杠杆
批处理	批处理令牌、作业、文件或已完成工作单元	把非交互式工作移出在线路径
微调	训练令牌 / 计算加模型托管	数据质量、实验次数、适配器策略
平台依赖项	搜索、存储、网络、遥测、安全、评估	架构、保留、采样、数据本地性

因此，令牌价格不等于单位经济性。价格更低的模型若需要更多重试、更长提示词、更大检索上下文或人工纠正，每个获接受业务结果的成本可能反而更高；更昂贵的模型若能一步完成工作流并避免工具故障，也可能更便宜。真正有意义的分母是结果：获接受的答案、已解决的工单、已审查的文档、成功的工具操作或完成的智能体任务。

模型资源契约应记录计费单位，以及提供商成本数据中可用的分配维度。FinOps 系统由此可以把用量连接到项目、应用、功能或租户。成本证据往往晚于请求遥测到达，因此对账应保留能够连接两条时间线的请求 ID 或聚合维度。

17.13 Azure 参考控制平面：Microsoft Foundry

Microsoft Foundry 为资源级视角提供了连贯参照。在云管理层，Foundry 资源通过 Microsoft.CognitiveServices 资源提供程序表示，账户种类为 AI Services。它治理安全、网络、模型部署、配额和计费。项目是组织开发、连接、智能体和评估的子资源，继承或使用父资源边界；模型部署也是账户下受治理的子资源。

表 17-14 Azure Foundry 参考对象

Azure 对象	控制平面责任	数据平面后果
Foundry 资源 / AI Services 账户	顶层身份、网络、计费、配额与部署边界	提供统一资源端点与授权上下文
Foundry 项目	应用、智能体、连接和评估的开发与协作边界	限定项目资产与用户体验的作用域；不会自动创建专用模型容量
模型部署	绑定部署名称、模型格式 / 名称 / 版本、SKU / 容量、升级行为与可选的负责任 AI 策略	在推理请求中提供部署名称
私有端点与 DNS	限制入站网络路径和名称解析	同一逻辑主机名从虚拟网络内部解析为私有地址
Azure RBAC 分配	在资源或项目作用域授予控制平面和 / 或数据平面权限	决定谁能部署、调用、运维或使用
Azure Monitor 与诊断	路由平台指标、日志与变更证据	支持 SLO、事件、容量与成本分析

部署资源公开模型格式、名称、版本、SKU、容量、负责任 AI 策略绑定和版本升级行为等字段。具体 API 版本和可用属性会演进，因此基础设施代码应以当前资源模板参考和所选区域进行验证。模型可用性也取决于区域和部署类型。

Foundry RBAC 区分管理责任和运行时责任。Microsoft 建议生产环境使用基于令牌的 Microsoft Entra 身份验证，而不是广泛共享的密钥。密钥仍支持兼容和初始使用，但与 Entra 身份相比，持有密钥不能提供相同的主体级归因或角色限制。

Azure 部署类型明确体现运营契约。全局、数据区、区域、标准、预配、批处理以及开发者或专用选项在处理作用域、容量、价格和工作负载适用性方面存在差异。这些名称因提供商而异；可迁移的经验是把部署类型视为架构。

17.14 多云转换：工程意图，而非功能对等

阿里云 Model Studio、腾讯云 TokenHub 和 TI-ONE，以及火山引擎 Ark，都公开了许多相同的工程关注点：模型目录、区域端点、API 凭据或 IAM、在线与批量推理、模型部署、配额、监控、用量统计、评估、微调和模型退役。这些对象与成熟度并不相同，因此比较应映射工程意图，而不是假定一一对应的产品对等性。

表 17-15 四个云平台的模型资源意图

工程意图	Azure	阿里云	腾讯云	火山引擎
顶层 AI 平台边界	Microsoft Foundry 资源与项目	Model Studio 账户 / 工作区与区域服务边界	TokenHub 与 TI-ONE 服务 / 项目边界	Ark 项目与平台资源
托管模型访问	Foundry Models 部署与统一推理端点	带区域与工作区端点的 Model Studio 模型服务	TokenHub 模型库与在线推理；TI-ONE 推理服务	Ark 在线推理与端点模式
身份与授权	Microsoft Entra ID、Azure RBAC、托管身份	阿里云 RAM 与工作区作用域凭据	腾讯云 CAM 与 API 密钥访问控制	火山引擎 IAM 与 API 密钥
容量与配额	订阅 / 账户 / 模型 / 部署类型配额与预配选项	区域 / 工作区并发、配额、在线与批处理契约	服务配额、推理服务配置、用量与警报	项目配额、端点配置、在线 / 批处理容量选项
监控与用量	Azure Monitor 指标、诊断、活动与成本数据	工作区用量、服务监控、错误与计费证据	用量统计、模型 / 推理监控、警报与部署日志	项目用量统计、推理端点监控与控制平面用量 API
生命周期	模型 / 版本部署、升级选项、弃用与退役	模型可用性、微调模型部署、区域端点生命周期	模型 / 推理服务生命周期与部署操作	模型部署、端点模式、已发布的退役通知

17.14.1 阿里云 Model Studio

Model Studio 提供带 OpenAI 兼容接口和提供商原生协议的一站式模型服务。区域边界在运营上很重要：不同区域有不同服务域名、API 密钥、可用模型与能力。生产环境可以使用工作区专用域名，以形成更清晰的工作区边界以及已记录的并发、网络与服务特性。微调模型、批处理作业、知识资源和智能体应用又让平台超越了原始推理。

17.14.2 腾讯云 TokenHub 与 TI-ONE

腾讯云把便捷模型访问与更广泛的模型工程分开。TokenHub 公开模型库、按量付费在线推理、可把访问范围限制到指定模型或推理服务的 API 密钥、用量统计与监控。TI-ONE 又加入模型开发和推理服务管理，包括部署日志与监控。这一组合说明，模型访问与 AI PaaS 生命周期是相关但不同的层。

17.14.3 火山引擎 Ark

火山引擎 Ark 组合模型推理、评估、微调和应用能力。其文档区分在线推理模式、批处理、低延迟选项、项目配额、API 密钥与 IAM、用量统计、端点安全监控和模型退役。即便产品术语不同，这些对象也支持同样的控制平面问题。

注意：云产品名称、模型目录、配额、价格、预览状态和区域可用性都容易变化。印刷章节应保留稳定的资源概念和决策问题；当前命令、模型名称与区域矩阵应放在配套存储库中，并在书稿冻结前重新验证。

17.15 模型部署决策记录

一份简短决策记录可把模型资源契约转化为运营制品。它应提交在基础设施代码旁，或存储在受治理的架构存储库中。该记录不能替代提供商状态；它解释状态为何存在，以及哪些证据授权变更。

示例 17-1：感知提供商的模型部署决策记录

```
apiVersion: cloudaipatterns.dev/v1
kind: ModelDeploymentRecord
metadata:
name: claims-assistant-primary
owner: claims-platform
environment: production
spec:
provider: azure
resourceId: /subscriptions/.../accounts/foundry-prod/deployments/claims-primary
project: claims-agent-prod
model:
name: <approved-model-family>
version: <approved-version>
format: OpenAI
deployment:
name: claims-primary
type: <deployment-type>
processingScope: <regional|data-zone|global>
upgradePolicy: explicit
access:
inferencePrincipal: <managed-identity-resource-id>
publicNetworkAccess: disabled
privateEndpoint: <resource-id>
safety:
guardrail: claims-prod-v3
objectives:
timeToFirstTokenP99: <threshold>
throttleRateMax: <threshold>
costPerAcceptedCaseMax: <threshold>
lifecycle:
evaluationSet: claims-golden-v5
rollbackDeployment: claims-previous
retirementOwner: claims-platform
```

在可能情况下，该记录应从实时状态生成或通过实时状态验证。手工复制的版本字符串与资源 ID 会发生漂移。平台流水线可以查询部署，与已批准记录比较，运行评估，并在模型版本、策略、区域或身份出现意外差异时阻止晋级。

能力检查 C4：检查并调用托管模型资源

这项独立能力检查把一个托管模型部署作为云资源进行验证，刻意避免构建完整应用。你可以使用现有 Microsoft Foundry 资源与模型部署，也可以使用所选区域当前支持的门户、CLI、Bicep 或 Terraform 工作流创建临时资源。由于模型可用性和配额会变化，本练习使用参数，而不指定某个商业模型。



图 17-6 能力检查生成模型资源证据包，而不只是一次成功提示词的截图。

C4.1 能力目标

证明一个部署能够被识别、由工作负载主体调用、接受模型与容量状态检查、关联到配额和遥测、评估网络与安全态势，并完成退役或明确指定所有者。主要交付物是 model-resource-evidence.json。

C4.2 前提条件与成本边界

- 拥有一个 Azure 订阅，并可在其中读取 Foundry 资源和模型部署。创建资源需要额外控制平面权限。

- Azure CLI 和 Python 3.11 或更高版本。为调用示例安装 azure-identity 与当前 openai 软件包。
- 拥有调用所选部署权限的 Microsoft Entra 主体。教学运行使用用户身份，自动运行使用托管身份 / 工作负载身份。
- 所选区域和配额池支持的模型部署。不要直接复制本书中的模型名称，而不检查当前可用性。
- 成本等级：在按用量部署上发起少量短推理请求时较低；预配或专用容量可能中等或较高。不要为本能力检查创建承诺容量。

C4.3 设置环境变量

示例 17-2：能力检查环境

```
export AZ_SUBSCRIPTION_ID="<subscription-id>"
export AZ_RESOURCE_GROUP="<resource-group>"
export AZ_AI_ACCOUNT="<foundry-resource-name>"
export AZ_DEPLOYMENT="<model-deployment-name>"

az account set --subscription "$AZ_SUBSCRIPTION_ID"
```

C4.4 检查账户与部署状态

调用模型之前先查询实时控制平面。API 使用的部署名称应解析到具有明确模型与 SKU 属性的可检查子资源。

示例 17-3：检查模型部署

```
az cognitiveservices account show \
--resource-group "$AZ_RESOURCE_GROUP" \
--name "$AZ_AI_ACCOUNT" \
--output json > account.json

az cognitiveservices account deployment show \
--resource-group "$AZ_RESOURCE_GROUP" \
--name "$AZ_AI_ACCOUNT" \
--deployment-name "$AZ_DEPLOYMENT" \
--output json > deployment.json

jq '{
deploymentName: .name,
model: .properties.model,
sku: .sku,
raiPolicyName: .properties.raiPolicyName,
versionUpgradeOption: .properties.versionUpgradeOption
}' deployment.json
```

具体属性集合取决于 API 版本和部署类型。保留未修改的 JSON 作为证据。汇总视图用于人工审查，不能替代提供商响应。

C4.5 记录身份与授权

识别将调用推理的主体，以及允许管理账户或部署的主体。不要从门户可见性推断运行时权限。记录账户与项目作用域的角色分配，并测试预期数据平面调用。

示例 17-4：在 AI 资源作用域捕获角色分配

```
ACCOUNT_ID=$(jq -r '.id' account.json)

az role assignment list \
--scope "$ACCOUNT_ID" \
--include-inherited \
--all \
--output json > role-assignments.json
```

C4.6 通过 Microsoft Entra ID 调用

以下 Python 示例使用持有者令牌提供程序，而不是长期 API 密钥。基址 URL 与令牌作用域反映当前 Foundry / OpenAI v1 模式；请对照所选资源的当前官方文档进行验证。

示例 17-5：无密钥推理调用

```
import json
import os
import time
from pathlib import Path
from azure.identity import DefaultAzureCredential, get_bearer_token_provider
from openai import OpenAI

account = os.environ["AZ_AI_ACCOUNT"]
deployment = os.environ["AZ_DEPLOYMENT"]

token_provider = get_bearer_token_provider(
DefaultAzureCredential(),
"https://ai.azure.com/.default",
)

client = OpenAI(
base_url=f"https://{account}.openai.azure.com/openai/v1/",
api_key=token_provider,
)

started = time.perf_counter()
response = client.responses.create(
model=deployment,
input="Return exactly: governed model resource verified",
max_output_tokens=32,
)
elapsed_ms = round((time.perf_counter() - started) * 1000, 1)

evidence = {
"responseId": response.id,
"deployment": deployment,
"elapsedMs": elapsed_ms,
"outputText": response.output_text,
"usage": response.usage.model_dump() if response.usage else None,
}
```

```
Path("inference-evidence.json").write_text(
json.dumps(evidence, indent=2)
)
print(json.dumps(evidence, indent=2))
```

成功响应证明运行时授权和端点可达，但不能证明最小权限。若教学环境允许，可使用同一主体尝试一项无害且预期被拒绝的管理操作，例如创建新部署或更改公共网络访问。保留拒绝结果，不要授予缺失的权限。

C4.7 检查网络、配额、遥测与安全证据

- 网络：记录 publicNetworkAccess、私有端点连接、DNS 假设与调用方网络路径。受身份保护的公共端点不等同于私有入口。
- 配额：记录提供商配额作用域、模型 / 版本或部署类型、已分配数量、当前用量和时间戳。仅有门户截图的证据较弱；若可用，应使用 API 或导出的记录。
- 遥测：列出 AI 资源的指标定义，并至少捕获该部署支持的请求数、令牌用量、延迟、失败与限流信号。
- 安全：记录绑定到部署的护栏或负责任 AI 策略，包括观测值究竟是显式策略还是平台默认值。
- 成本：记录计费单位与资源可用的成本分配维度。测试请求后，成本数据可能不会立即可用。

示例 17-6：捕获指标定义与网络态势

```
az monitor metrics list-definitions \
--resource "$ACCOUNT_ID" \
--output json > metric-definitions.json

# Inspect the live account state used for network evidence.
jq '{
id: .id,
location: .location,
kind: .kind,
publicNetworkAccess: .properties.publicNetworkAccess,
privateEndpointConnections: .properties.privateEndpointConnections
}' account.json > network-evidence.json
```

C4.8 组装证据包

示例 17-7：组装感知提供商的证据包

```
import json
from pathlib import Path

def load(path):
return json.loads(Path(path).read_text())

account = load("account.json")
deployment = load("deployment.json")
roles = load("role-assignments.json")
network = load("network-evidence.json")

packet = {
"schema": "cloudaipatterns.dev/model-resource-evidence/v1",
"capturedAt": "<UTC timestamp>",
"resource": {
"id": account["id"],
"location": account["location"],
"kind": account.get("kind"),
},
"deployment": {
"name": deployment["name"],
"model": deployment.get("properties", {}).get("model"),
"sku": deployment.get("sku"),
"guardrail": deployment.get("properties", {}).get("raiPolicyName"),
"upgradeOption": deployment.get("properties", {}).get("versionUpgradeOption"),
},
"authorization": {"roleAssignments": roles},
"network": network,
"runtimeEvidence": load("inference-evidence.json"),
"quotaEvidence": "<attach current provider export or API result>",
"telemetryEvidence": "metric-definitions.json",
"billingUnit": "<record current deployment billing contract>",
"retirementOwner": "<team or named role>",
}
```

```
Path("model-resource-evidence.json").write_text(
json.dumps(packet, indent=2)
)
```

C4.9 验证标准

表 17-16 能力检查验收标准

标准	通过证据
资源身份	提供商资源 ID、账户、项目（如使用）、区域、部署名称
模型身份	模型格式、名称、版本及部署类型 / SKU
运行时授权	带响应 / 请求 ID、通过 Entra 验证身份的成功请求
最小权限	已记录的角色作用域，以及在安全情况下的一项预期被拒绝的管理操作
网络态势	公共 / 私有访问状态、适用时的私有端点证据、调用方路径
配额 / 容量	作用域、单位、分配或部署容量、时间戳
安全绑定	具名护栏 / 负责任 AI 策略，或已记录的平台默认值
遥测	受支持的指标定义，以及传播后观测到的一项请求 / 用量信号
计费	计费单位与成本分配作用域；注明预期的报告延迟
生命周期	升级行为、弃用所有者及清理或保留决策

C4.10 清理与保留

如果创建了临时部署，请删除该部署，并验证其不再出现在部署列表中。如果只为本能力检查创建了专用资源组，请在导出证据包后删除该资源组。不要仅因练习完成就删除共享账户。保留证据包，但不得保留原始提示词、秘密或访问令牌。

示例 17-8：仅删除临时部署并验证退役

```
az cognitiveservices account deployment delete \
--resource-group "$AZ_RESOURCE_GROUP" \
--name "$AZ_AI_ACCOUNT" \
--deployment-name "$AZ_DEPLOYMENT"

az cognitiveservices account deployment list \
--resource-group "$AZ_RESOURCE_GROUP" \
--name "$AZ_AI_ACCOUNT" \
--output table
```

17.16 CloudDesk 复用：安全的模型能力层

CloudDesk FinOps Agent 以后可能使用模型服务，但本书不实现其运行时或租户系统。本章可复用的输出是一份能力契约，允许智能体或平台服务检查并调用模型，而不把端点当作不透明字符串。

表 17-17 面向 CloudDesk 的能力契约

能力	用途	所需控制
llm.deployment.inspect	返回模型、版本、部署类型、区域、护栏、容量、升级与退役元数据	只读云身份；不可访问提示词或响应
llm.inference.invoke	在工作负载身份和有界请求预算下调用已批准部署	部署允许列表、令牌限制、相关 ID、安全与遥测绑定
llm.quota.read	读取当前配额作用域、分配、利用率与限流证据	只读配额访问；时间戳与提供商作用域
llm.retirement.assess	查找受弃用影响的部署，并提出迁移计划	清单、提供商退役信息流、依赖图、评估门禁；不得自动执行破坏性操作

这些能力应返回结构化证据，而不只是自然语言。llm.deployment.inspect 绝不能泄露 API 密钥；llm.inference.invoke 应要求已批准的部署名称，而不接受提示词提供的任意端点；llm.retirement.assess 可以建议替代方案，但不得在未经变更批准和依赖验证的情况下删除部署。

17.17 要点

- 模型 API 是数据平面协议。模型资源加入所有权、身份、隔离、配额、安全、遥测、计费、变更与退役所需的控制平面状态。

- 模型家族、模型版本、部署、端点、项目和 AI PaaS 资源是不同对象，在架构与证据中都应保持区分。
- MaaS 是不可或缺的托管推理层，但 AI PaaS 还治理项目、知识、评估、智能体、工具、身份、可观测性与生命周期。
- 部署类型就是架构：共享、区域、数据区、预配、批处理、专用和微调模式编码不同的容量、延迟、本地性与成本契约。
- 生产工作负载应优先使用工作负载身份和限定作用域的授权，而不是广泛共享的长期 API 密钥。
- 配额不等于吞吐量，吞吐量不等于 SLO，预算也不是硬性容量控制。每项都需要各自的证据和响应机制。
- 模型升级与退役是发布事件。稳定部署名称有利于调用方，但实际模型版本与策略版本必须保持可见。
- 护栏是版本化控制，用于补充而非取代身份、授权、网络、验证与人工批准。
- 推理运维需要相互关联的请求、令牌、延迟、限流、安全、变更与成本证据，并采用保护隐私的内容采集方式。
- 多云比较在映射工程意图而不是强求表面功能对等时最有价值。

参考文献

[1] 美国国家标准与技术研究院：《NIST 云计算定义》（SP 800-145）。https://doi.org/10.6028/NIST.SP.800-145访问于 2026 年 8 月 24 日。

[2] 美国国家标准与技术研究院：《零信任架构》（SP 800-207）。https://doi.org/10.6028/NIST.SP.800-207访问于 2026 年 8 月 24 日。

[3] 美国国家标准与技术研究院：《AI 风险管理框架》（AI RMF 1.0）。https://doi.org/10.6028/NIST.AI.100-1访问于 2026 年 8 月 24 日。

[4] Microsoft Learn：Microsoft Foundry 架构。https://learn.microsoft.com/en-us/azure/foundry/concepts/architecture访问于 2026 年 8 月 24 日。

[5] Microsoft Learn：Microsoft Foundry 的基于角色访问控制。https://learn.microsoft.com/en-us/azure/foundry/concepts/rbac-foundry访问于 2026 年 8 月 24 日。

[6] Microsoft Learn：Microsoft Foundry Models 的部署类型。https://learn.microsoft.com/en-us/azure/foundry/foundry-models/concepts/deployment-types访问于 2026 年 8 月 24 日。

[7] Microsoft Learn：Microsoft Foundry Models 中 Azure OpenAI 的配额与限制。https://learn.microsoft.com/en-us/azure/foundry/openai/quotas-limits访问于 2026 年 8 月 24 日。

[8] Microsoft Learn：为 Microsoft Foundry 配置专用链接。https://learn.microsoft.com/en-us/azure/foundry/how-to/configure-private-link访问于 2026 年 8 月 24 日。

[9] Microsoft Learn：Microsoft Foundry 护栏概览。https://learn.microsoft.com/en-us/azure/foundry/guardrails/guardrails-overview访问于 2026 年 8 月 24 日。

[10] Microsoft Learn：Microsoft.CognitiveServices accounts/deployments 资源参考。https://learn.microsoft.com/en-us/azure/templates/microsoft.cognitiveservices/accounts/deployments访问于 2026 年 8 月 24 日。

[11] Microsoft Learn：使用 Bicep 模板创建 Microsoft Foundry 资源与项目。https://learn.microsoft.com/en-us/azure/foundry/how-to/create-resource-template访问于 2026 年 8 月 24 日。

[12] Microsoft Learn：Azure CLI 认知服务账户部署命令。https://learn.microsoft.com/en-us/cli/azure/cognitiveservices/account/deployment访问于 2026 年 8 月 24 日。

[13] Microsoft Learn：使用聊天模型和 v1 API。https://learn.microsoft.com/en-us/azure/foundry/openai/how-to/chatgpt访问于 2026 年 8 月 24 日。

[14] Microsoft Learn：Microsoft.CognitiveServices 账户支持的指标。https://learn.microsoft.com/en-us/azure/azure-monitor/reference/supported-metrics/microsoft-cognitiveservices-accounts-metrics访问于 2026 年 8 月 24 日。

[15] Microsoft Learn：Azure Monitor 文档。https://learn.microsoft.com/en-us/azure/azure-monitor/访问于 2026 年 8 月 24 日。

[16] 阿里云文档：《什么是 Model Studio？》。https://www.alibabacloud.com/help/en/model-studio/what-is-model-studio访问于 2026 年 8 月 24 日。

[17] 阿里云文档：Model Studio 区域与端点。https://www.alibabacloud.com/help/en/model-studio/regions/访问于 2026 年 8 月 24 日。

[18] 阿里云文档：Model Studio 中的批量推理。https://www.alibabacloud.com/help/en/model-studio/batch-inference访问于 2026 年 8 月 24 日。

[19] 阿里云文档：Model Studio 错误码。https://www.alibabacloud.com/help/en/model-studio/error-code访问于 2026 年 8 月 24 日。

[20] 腾讯云文档：TokenHub 快速入门。https://www.tencentcloud.com/document/product/1300/78939访问于 2026 年 8 月 24 日。

[21] 腾讯云文档：TokenHub API 密钥管理。https://www.tencentcloud.com/document/product/1300/78949访问于 2026 年 8 月 24 日。

[22] 腾讯云文档：TI-ONE API 类别。https://www.tencentcloud.com/document/product/851访问于 2026 年 8 月 24 日。

[23] 火山引擎文档：火山引擎 Ark 文档。https://docs.volcengine.com/docs/82379访问于 2026 年 8 月 24 日。

[24] 火山引擎文档：Ark 推理模式。https://www.volcengine.com/docs/82379/2123245访问于 2026 年 8 月 24 日。

[25] 火山引擎文档：Ark 低延迟在线推理。https://www.volcengine.com/docs/82379/2335857访问于 2026 年 8 月 24 日。

[26] OpenTelemetry：生成式 AI 系统的 OpenTelemetry 语义约定。https://opentelemetry.io/docs/specs/semconv/gen-ai/访问于 2026 年 8 月 24 日。

[27] OWASP 基金会：大型语言模型应用十大风险。https://owasp.org/www-project-top-10-for-large-language-model-applications/访问于 2026 年 8 月 24 日。

[28] Google：《SRE 工作簿——实施 SLO》。https://sre.google/workbook/implementing-slos/访问于 2026 年 8 月 24 日。

[29] FinOps 基金会：FinOps 框架。https://www.finops.org/framework/访问于 2026 年 8 月 24 日。

[30] FinOps 基金会：FOCUS 规范。https://focus.finops.org/访问于 2026 年 8 月 24 日。

第五篇 / 第 18 章

RAG 不是向量数据库

把知识作为云资源进行治理

生产级检索系统并非由向量存在哪里定义，而是由知识如何获取、转换、授权、版本化、检索、举证、评估、观测和退役来定义。

本章内容

- 为何向量索引对生产级 RAG 必要但不充分
- 如何把知识建模为具有明确生命周期的受治理云资源
- Azure AI Search 与 Foundry IQ 如何分离搜索管理对象和面向智能体的知识对象
- 何时使用词法、向量、混合、语义、筛选和智能体式检索
- 身份、网络、授权、版本控制、证据、评估、可观测性与成本如何纳入控制平面
- 如何创建、评估、版本化、晋级、回滚和清理独立的受治理知识库
- 同一工程意图如何映射到阿里云 Model Studio、腾讯云 ADP，以及火山引擎 AgentKit 和 Ark

版本说明

本章的产品名称、API 成熟度与可用性已对照 2026 年 8 月 24 日的官方文档核验。Azure 实验的稳定路径使用 REST API 版本 2026-04-01。仍处于预览的能力均明确标注，且核心实验不依赖这些能力。

18.1 只有四行、直到关键时刻才失效的演示

检索增强生成原型可能小得令人产生错觉：取一组文档，将其分块，把每个分块转换为嵌入，把向量放入索引，检索与问题最接近的邻项，再把这些邻项附加到提示词。最初的 RAG 架构确立了把参数化语言模型与外部非参数记忆相结合的强大能力。这一基本模式仍然成立；生产环境中改变的是“知识”一词附带的责任数量。[1]

因此，第一次成功回答是一个危险的里程碑。它证明语义检索在一套配置下能够处理一个数据集，却没有证明知识是最新的、已授权的、可追溯的、可复现的、成本受控的，或能被安全替换。演示回答的是：模型能否找到有用段落？生产系统还必须回答：来源由谁负

责？哪个转换生成了这个分块？哪些身份可以检索它？相较昨天发生了什么变化？哪个索引版本生成了答案？结果能否复现？发布能否回滚？检索花费多少？审计人员可以检查哪些证据？

吸引人的原型

```
documents = load_files("./handbook")
chunks = split(documents)
vectors = embed(chunks)
index.upsert(vectors)

context = index.nearest(embed(user_question), k=5)
answer = llm.generate(user_question, context=context)
```

这段代码本身没有错。问题在于，每个被省略的关注点都会在其他地方重新出现：临时同步脚本、硬编码筛选器、应用缓存、记录来源所有者的电子表格、手工重建索引流程、安全审查或事件报告。平台可以隐藏这些责任，却无法消除它们。

向量数据库是为相似性搜索优化的数据结构与查询服务。受治理知识资源则是一份托管契约，把源内容、转换、检索行为、授权、证据、质量、运维和生命周期连接起来。向量搜索只是该契约内的一种机制。把它当成整个系统，就像把生产数据库应用称作一棵 B 树。



图 18-1 向量索引只是受治理知识生命周期中的一个组件。

控制平面规则

如果知识变更无法通过明确管理路径完成授权、版本化、观测、评估、晋级和逆转，系统就还没有生产级知识控制平面。

18.1.1 生产差距

通过故障模式最容易看清生产差距。假设对象存储中的策略文档已被替换，但索引器尚未运行，答案在语法上流畅，却在运营上过期。假设索引包含租户或部门元数据，但应用有一个端点忘记应用筛选器，答案在语义上相关，却未获授权。假设嵌入部署在未协调索引

迁移的情况下更改维度，摄取流水线会失败，或悄然指向错误索引。假设新的分块策略提升平均相关性，却删除某项受监管程序所需的段落；没有回归集与回滚路径，团队只能从用户投诉中发现损失。

这些不是向量距离问题，而是生命周期问题；同时也是云控制平面问题，因为它们需要具名资源、身份、策略、网络边界、部署记录、指标、日志、配额、预算与状态转换。云的价值不只是托管存储，而是让知识操作变得明确、可自动化且可治理。

18.2 从向量索引到知识资源

在本书中，知识资源由九个相互耦合的元素定义：

K = <S, T, X, R, A, V, E, O, C>

- S——来源与出处：准入什么内容、由谁负责，以及来自何处
- T——转换：解析、规范化、分块、增强、嵌入和元数据提取
- X——索引与架构：可搜索契约，包括文本、向量、来源字段和安全元数据
- R——检索策略：词法、向量、混合、筛选、重排序、查询改写和运行时限制
- A——授权与访问边界：身份、角色、文档级控制、网络路径和秘密
- V——版本控制与发布状态：候选、当前、已退役、回滚目标和兼容性预期
- E——证据与评估：分块、引用、活动、测试集、质量阈值和故障分析
- O——运维：新鲜度、延迟、可用性、摄取运行状况、事件响应和清理
- C——成本与容量：存储、摄取、嵌入、检索、重排序、模型使用、配额与预算

这个元组并不是新的 API，而是一项完整性测试。团队可以使用托管搜索服务、云 AI 平台、开源组件或其组合实现这些元素。工程问题在于这些元素是否有明确的所有者与接口。如果向量索引受管理，而来源准入、访问筛选、评估和回滚仍然隐含，知识系统就只接受了部分管理。

关注点	向量数据库视角	知识资源视角
主要对象	集合或索引	具有来源、检索、访问和证据契约的版本化知识产品
输入	向量与元数据	已授权来源加受治理转换
查询	最近邻搜索	组合式检索策略：词法、向量、筛选、重排序、限制与引用
安全	凭据和可选元数据筛选器	身份、网络边界、服务器注入的授权、来源权限与审计
变更	更新插入或重建	候选构建、评估、晋级、兼容性检查、回滚与退役

关注点	向量数据库视角	知识资源视角
质量	相似度分数	检索覆盖率、排序、依据性、引用有效性、安全排除、延迟与成本
运维	容量与查询指标	摄取运行状况、新鲜度、检索追踪、SLO、事件响应与血缘

18.2.1 资源级测试

当一项能力可以独立于使用它的应用代码进行寻址和治理时，它就达到资源级。实际测试不在于门户是否显示一个磁贴，而是要问该能力是否有持久身份、生命周期、授权模型、自动化接口、运营遥测与可问责的成本边界。

按此测试，本地嵌入数组不是知识资源。托管索引可能是一项资源，但仍不是完整知识资源。只有当来源与索引能够登记、检索行为能够配置、访问条件能够施加、结果能够返回引用与活动，且变更可以晋级或逆转而无需重写每个使用方智能体时，生产知识库才有完整意义。

设计含义

应用和智能体应依赖稳定的知识契约，而不是文件路径、原始向量集合名称或供应商门户工作流。

18.3 知识生命周期

18.3.1 有意识地准入来源

来源不只是位置，还承载所有权、权威性、保留、敏感度与预期更新行为。即使两份 PDF 文本完全相同，若一份是已批准策略，另一份是过时草稿，其运营含义也不同。因此，来源注册表至少应记录稳定来源标识符、所有者、规范 URI、内容类型、司法管辖区或区域、敏感度、生效日期、过期或复审日期，以及控制准入的策略。

这是 RAG 最早可能静默失效的地方。团队往往先摄取最容易上传的内容，再要求模型调和矛盾。受治理设计反其道而行：在检索之前解决来源权威性，并保留足够来源信息以解释一段内容为何合格。

18.3.2 让转换可观测

解析、OCR、章节检测、表格提取、分块、标题生成、元数据增强和嵌入都是转换，每一步都可能改变内容含义或可检索性。因此，分块是派生制品，而不是原始文档；其记录应识别产生它的来源版本与转换版本。

托管索引器与技能集很有价值，因为它们把反复出现的摄取工作转化为具名、可检查对象。拉取式索引器可以按计划运行，使用数据源连接，调用增强技能并写入目标索引。其价值不在于代码消失，而在于计划、状态、字段映射、重试行为和失败证据成为平台状态的一部分。[7]

18.3.3 把架构视为应用契约

索引架构决定应用能够搜索、筛选、返回、引用和保护什么。有用的最低分块架构包含稳定键、内容、标题、来源 URI、来源标识符、来源版本、时间戳、内容哈希、租户或工作区标识符、访问控制标识符，以及一个或多个向量字段。用于筛选的字段必须声明为 filterable；向用户展示或传给模型的字段必须是 retrievable；用于全文搜索的字段必须是 searchable。

架构错误代价高昂，因为有些属性不能原地更改。例如，在 Azure AI Search 中，不能直接把已有字段改成 filterable，必须重建或替换索引。这让看似微小的元数据决策变成版本控制问题。[5]

18.3.4 将授权绑定到检索

必须在未授权内容成为模型上下文之前执行检索授权。检索后在用户界面隐藏引用是不够的；一旦未授权分块进入提示词，保密边界就已经失效。

应用应从经过身份验证的服务器端上下文派生筛选器。用户或模型可以提供业务查询，但都不应构造最终安全谓词。tenantId、workspaceId、departmentId、documentAcl、sensitivity 和 effectiveDate 等元数据属于检索契约。精确筛选与语义相关性用途不同：相关性分数回答“什么相似”，筛选器回答“什么合格”。

18.3.5 版本化、评估、晋级与退役

分块、嵌入、分析器、语义配置和来源选择都可能改变检索行为。这些变化应创建候选知识版本。候选版本装载后，应针对稳定问题集、安全测试与运营阈值进行评估，然后才能晋级为当前版本。上一版本应保留足够时间，以支持回滚和事件分析。

退役也是控制平面操作。删除应用不一定会删除索引、来源连接器、模型部署、日志或存储文档。受治理生命周期会定义删除顺序、证据保留，以及知识资源不再产生成本的时间点。

18.4 以 Azure 作为参考对象模型

Azure AI Search 是有用的参照，因为它同时公开成熟的搜索管理层和面向智能体的知识层。搜索管理层包含索引、文档、数据源、索引器、技能集、同义词映射和别名。Foundry IQ 建立在 Azure AI Search 之上，向智能体提供可复用、感知权限的知识库。知

识库引用一个或多个知识源并提供检索行为；retrieve 操作可以返回内容、引用和活动。[2]

重要的经验不是产品名称，而是责任分离。搜索管理对象治理内容如何变得可搜索；知识对象治理智能体如何使用可搜索内容。它们是同一生命周期的两个视角。

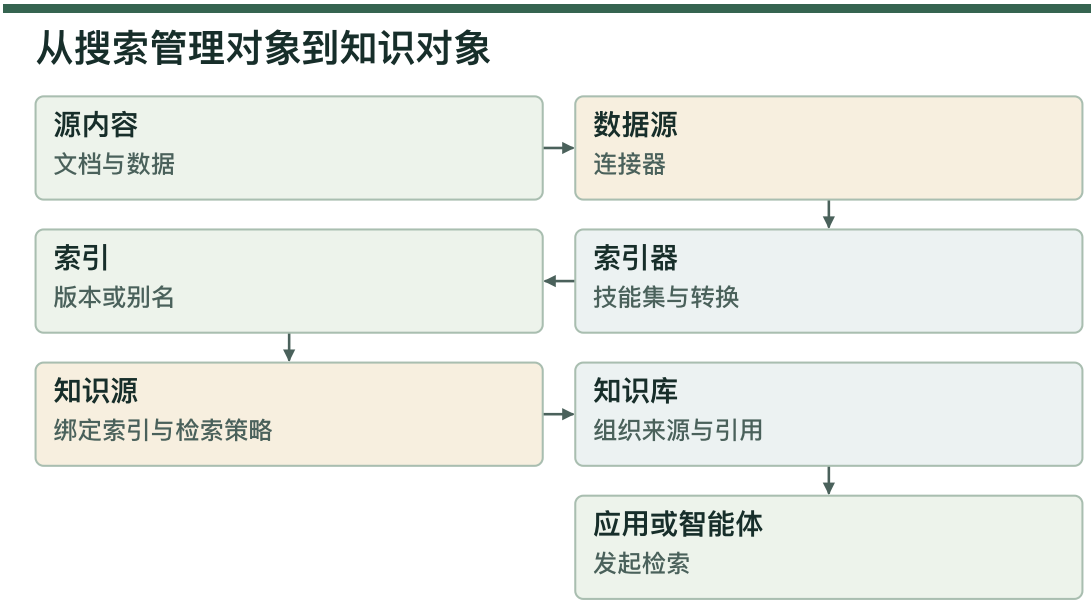


图 18-2 搜索管理对象为面向智能体的知识对象供给内容。

对象	控制平面责任	数据平面效果
搜索服务	容量、网络、身份、加密、诊断、配额	托管索引并执行查询
数据源	拉取式摄取的连接与来源作用域	向索引器提供原始内容
索引器	计划、映射、状态、错误处理、目标	运行摄取并写入文档
技能集	可复用的增强图与模型依赖项	解析、分块、增强并向量化
索引	架构、分析器、向量配置、语义配置	存储可搜索文档与向量
别名	映射到一个具体索引的稳定辅助名称	让受支持客户端查询或编制索引时无需硬编码版本
知识源	封装搜索索引、Blob 来源、OneLake 来源或 Web 来源以供检索	提供来源特定的检索界面

对象	控制平面责任	数据平面效果
知识库	组合知识源与默认检索行为	为应用或智能体执行 retrieve 操作
retrieve 响应	运行时选项，例如筛选器、引用与活动	返回依据内容、引用、执行活动、延迟和部分错误

18.4.1 控制平面与数据平面

控制平面创建并配置资源，数据平面移动内容并回答查询。该边界对安全和运维很重要。部署身份可以创建索引或更新知识库，而应用身份只能从已批准知识库检索。不能仅因管理员密钥同时能执行两种功能，就混淆这两个身份。

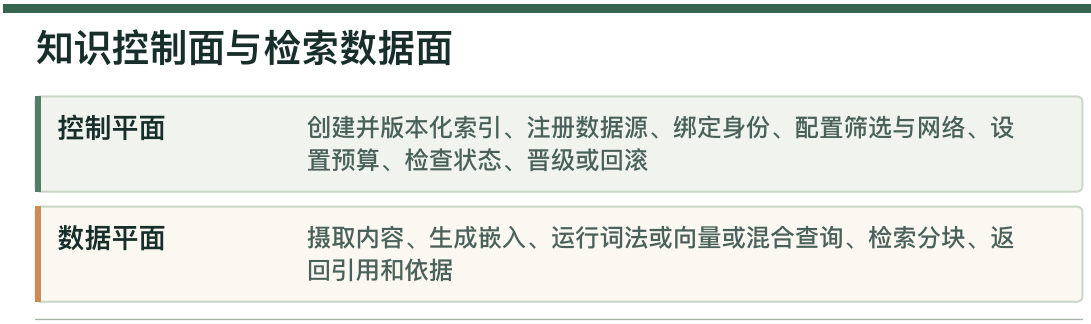


图 18-3 知识控制平面治理检索数据平面。

当前 Azure 状态——2026 年 8 月 24 日

2026-04-01 REST API 为正式版知识源类型（包括搜索索引知识源）提供正式版抽取式检索，索引别名也已正式发布。查询规划、答案综合、可配置推理强度和部分较新集成仍是预览功能，需要预览版 API。核心实验使用稳定的 2026-04-01 路径；预览功能仅作为扩展讨论。[3]

18.4.2 知识源与知识库

知识源为底层来源赋予面向检索的身份。在稳定 API 中，搜索索引知识源指定具体搜索索引、要搜索的字段、引用中要包括的来源字段，以及可选语义配置。Azure 还为 Blob Storage、已编制索引的 OneLake 和 Web 内容定义知识源类型。[8]

知识库引用一个或多个知识源。retrieve API 接受运行时参数，例如来源特定筛选器、重排序阈值、是否包括引用、最大运行时间、最大输出大小和是否返回活动。响应可以携带检索内容、引用、搜索参数、已用时间、令牌信息和每来源错误。这是重要的架构转变：检索成为类型化服务交互，而不是不透明辅助函数。[9][10]

18.5 检索是一项策略，而不是单一算法

没有一种通用检索算法可以主导每种企业查询。产品代码、法律条款编号、日期、名称和不常见技术术语适合精确匹配；释义和概念相关语言适合向量相似性；安全边界要求精确资格检查；复杂问题可能需要分解成多次搜索。因此，生产设计应表达检索策略，而不是选择时髦的存储引擎。

18.5.1 词法检索

词法检索搜索倒排索引并对文本匹配进行排序，通常使用 BM25。当查询含有必须出现在来源中的辨别性术语时，它很有效，例如错误码、SKU、法规编号、API 名称或引号内短语。其弱点是词汇不匹配：用户可能询问成本归因，而来源使用内部结算和分配等说法。

18.5.2 向量检索

向量检索比较查询嵌入与存储嵌入，即使措辞不同也能检索语义相关段落。其弱点包括近似邻居行为、依赖嵌入模型、对分块敏感、处理精确标识符能力较弱，以及把高语义相似度误当成授权或事实权威的风险。

18.5.3 混合检索与 RRF

混合检索并行执行全文查询与向量查询，再合并排序列表。Azure AI Search 使用倒数排序融合（RRF）进行合并。混合搜索很有吸引力，因为精确证据与语义证据可以相互增强，而无需假装它们的原始分数可以直接比较。Microsoft 建议使用真实数据测试；标识符和专业术语通常受益于词法搜索，向量搜索则捕获概念相似性。[4]

RRF 组合排序位置，而不是原始分数。简化来说，每个结果会从其出现的每份排序列表获得 1/(k + rank) 的贡献。在两份列表中都排名靠前的文档，比只出现在一份列表中的文档积累更多证据。常数 k 降低最顶端位置的主导程度。确切生产实现因平台而异，但原则稳定：融合独立排序的证据，而不是凭直觉归一化不相关的分数尺度。

18.5.4 筛选器决定资格，而非相关性

筛选器作用于标记为 filterable 的非向量字段，可以按租户、工作区、部门、语言、日期、敏感度、产品、地域或其他精确属性进行限制。Azure 文档明确把安全筛选列为基础用例：把请求方的安全标识符与索引字段匹配。[5]

这种区别应体现在代码审查清单中。绝不能因为租户标识符在嵌入空间中碰巧不相似，就相信向量查询能隔离租户 A 与租户 B。授权谓词必须由可信应用构造，并应用到搜索请求或知识源运行时参数。

18.5.5 语义重排序

语义排序器作用于初始候选集，使用更深层语言理解重新排序结果或提取说明文字。它通常是第二阶段操作：词法与向量检索创建候选，重排序器把额外计算花在最有希望的候选上。这可以改善答案依据性，却引入延迟、成本和配置依赖。应把它作为发布变量评估，而不是只因门户提供开关就启用。

18.5.6 智能体式检索

智能体式检索在策略中加入查询规划、多项子查询、来源选择、合并和结构化证据。它适合复合问题：比较两项策略、识别例外，再解释对特定部门的影响。代价是增加模型使用、检索操作、延迟和故障面。

截至本章核验日期，Azure 公开了稳定的抽取式 retrieve 路径，以及更完整查询规划和答案综合的预览能力。这是有用的生产区别。团队可以先采用稳定知识资源契约，再仅在问题集确有需要时加入智能体式规划。

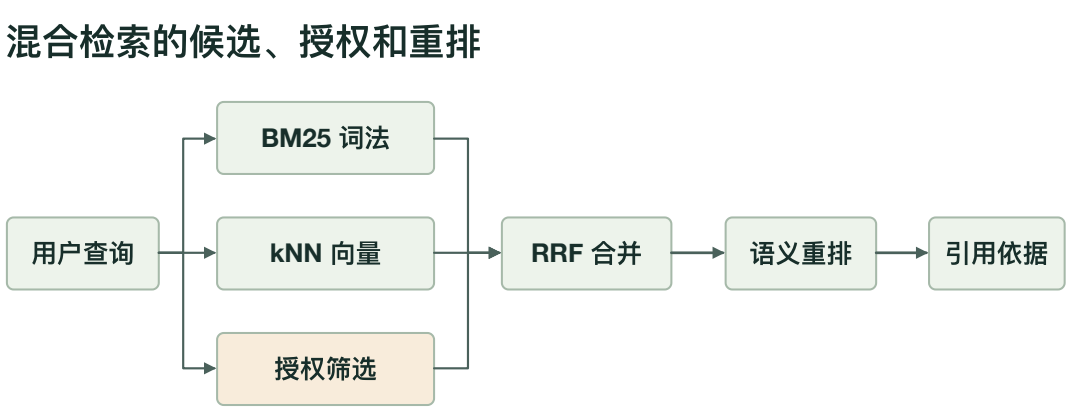


图 18-4 稳健的检索策略组合相关性、资格、重排序与证据。

问题模式	建议的起始策略	原因
精确标识符、名称、日期、条款、错误码	词法加精确筛选	精度与精确匹配占主导
对已批准内容进行概念释义	向量加授权筛选器	词汇不匹配占主导；资格仍须精确
一般企业问答	混合检索加授权筛选器；评估语义重排序	平衡精确召回与语义召回
具有多项约束的复合问题	使用带显式意图的稳定检索，或在评估成本 / 延迟后使用智能体式检索	可能需要多次搜索

问题模式	建议的起始策略	原因
需要审计的高风险答案	可用任何相关策略，但必须返回引用与活动	证据与可复现性优先于便利性
延迟或成本敏感端点	使用不含智能体规划的词法或混合检索；谨慎使用阈值和缓存	避免不必要的模型与查询扇出

18.6 治理与安全

18.6.1 身份与最小权限

在平台允许的情况下，应分别为部署、摄取、检索和运维使用不同身份。部署流水线可以创建或更新索引与知识对象；摄取身份只能读取已批准来源位置并写入目标索引；应用身份可以检索，但不能更改架构；运维身份可以读取状态、指标和日志，但不能读取文档内容，除非事件处置流程要求。

API 密钥适合实验，但生产设计应在受支持时优先采用工作负载身份和基于角色的授权。密钥轮换、秘密存储和客户管理的加密密钥属于控制平面。Azure 知识源与知识库定义除了支持 API 密钥，也支持 OAuth；在合格服务上，其定义还可以由客户管理密钥保护。[8][9]

18.6.2 私有连接

知识系统横跨多条网络路径：应用到搜索、索引器到来源、搜索到嵌入或增强模型，以及管理员到控制平面。仅限制查询端点会让摄取路径保持暴露。私有设计会盘点每项出站与入站依赖，在受支持时建立私有端点或受控出站，配置名称解析，并测试公共访问确已遭到拒绝。

验证步骤十分重要。门户中显示私有端点，并不能证明应用正在使用它。请从获准网络测试，再从未获准网络测试，检查 DNS 解析，并把结果保留为部署证据。

18.6.3 文档级资格

常见的共享索引架构包含 tenantId、workspaceId、allowedGroupIds、sensitivity、effectiveFrom、effectiveTo 与 status。具体字段各异，但三条规则稳定不变：

1. 可筛选的安全字段由可信摄取过程填充，不能盲目复制不可信文档文本。

2. 最终授权筛选器由经过身份验证的服务器上下文派生，不能由语言模型提供，也不能把用户输入作为原始语法接受。

3. 检索测试套件应包含反向用例，证明未授权文档不会出现在返回分块和模型可见上下文中。

平台提供的权限感知检索可以减少自定义逻辑，却不会免除理解边界的需要。例如，当前 Foundry IQ 文档说明了对受支持来源采用权限感知知识库和调用方身份强制执行。某些高级权限集成仍依赖特定来源或处于预览状态。应把它们作为待验证能力，而不是继承的假设。[2][3]

安全边界

生成后脱敏不能替代检索前授权。模型不得接收调用方无权检索的内容。

18.6.4 来源与引用

只有当引用可以解析到稳定、已授权来源并识别实际使用的证据时，它才有用。应存储规范来源 URI 或文档标识符、来源版本或内容哈希、分块标识符、标题以及相关位置元数据。生成的答案应保留声明与检索引用之间的映射。

这一设计改善的不只是用户信任，还支持事件分析、过期来源检测、来源退役和评估。用户质疑答案时，团队可以检查确切分块与检索活动，而不必尝试从日志中重建不可见提示词。

18.7 运维：新鲜度、证据、SLO 与成本

18.7.1 新鲜度是明确目标

新鲜度不等同于摄取成功。应定义从权威来源变化到可供搜索的最大可接受延迟。策略知识库可能要求一小时的新鲜度目标，归档研究语料库则可以接受每日计划。该目标决定索引器计划、事件触发器、重试策略和警报阈值。

同时监控聚合流水线状态和文档级故障。即使部分文档解析失败、超过大小限制或违反架构假设，流水线仍可能报告成功。当前 Azure 知识源状态 API 包含正式版知识源类型的同步状态和文档级错误详情。[3]

18.7.2 返回证据信封

检索结果应建模为证据信封，至少包括依据分块、引用、已应用筛选器或策略版本、检索活动、延迟与错误状态。对托管智能体式检索，还可能包含令牌和模型活动。Azure 稳定版 retrieve 响应可以包括引用、来源数据、重排序分数、搜索参数、已用时间和部分每来源错误。[10]



图 18-6 仅有文本并不足以构成生产级检索结果。

18.7.3 定义检索 SLO

有用的检索 SLO 应从应用视角撰写。例如：

- 新鲜度：99% 的已批准来源更新在 60 分钟内可供搜索。
- 可用性：99.9% 的检索请求在月度窗口内返回完整响应或明确标记为部分响应。
- 延迟：95% 的标准混合检索请求在 900 毫秒内完成，不包括答案生成。
- 引用完整性：100% 的已显示引用可解析到已授权来源版本。
- 安全排除：持续隔离测试中返回的未授权哨兵文档数为零。
- 质量：发布问题集中至少 90% 的问题能在前五个结果中检索到已批准支持段落。

这些数值是示例，不是通用目标。其价值在于把知识运维转化为发布与事件纪律。新索引版本未达到质量阈值时不能晋级；新鲜度耗尽错误预算时，应暂停接入新来源，直到恢复摄取可靠性。

18.7.4 计算完整检索成本

RAG 成本不只是向量存储，还可能包括来源存储、解析与 OCR、分块、嵌入、索引容量、副本与分区、语义重排序、查询规划、答案综合、网络传输、日志和模型令牌。相关单位通常是每个成功有依据答案的成本，而不是每次查询的成本。

因此，成本工作表应记录每文档或每 MB 的摄取成本、每请求检索成本、规划或综合使用的模型令牌、存储增长、保留版本，以及运营系统所需遥测的成本。成本也是检索策略输入：智能体式分解可能改善困难问题，却会浪费在精确查找上。

18.7.5 版本控制与回滚

索引别名提供映射到某个具体索引的稳定辅助名称，并可为受支持的文档与查询操作进行更新，适合蓝绿索引替换。然而，Azure 目前不允许把别名用作索引器目标，也不允许知识源引用别名。这项限制会改变晋级设计：直接搜索客户端可以使用别名，而知识源绑定必须显式更新到新的具体索引。[6]



图 18-5 直接搜索别名晋级与知识源晋级相关但不同。

18.8 评估是资源生命周期的一部分

RAG 评估应分离检索与生成。如果从未检索到正确段落，提示词变更无法弥补缺失证据；如果已检索该段落，但模型忽略或反驳它，故障属于生成或指令设计。RAGAS 形式化了忠实度、答案相关性和上下文相关性等无参考指标，后续研究又扩展了评估范围。更广泛的经验是分别测试模块与端到端行为。[11]

18.8.1 最低评估集

对于小型知识资源，应从十到三十个有意选择的问题开始，而不是数千条合成提示词。每个用例应包括预期来源或段落、允许和禁止的身份、关键精确术语，以及问题为何重要。还应包括简单、模糊、复合、过期、超出范围和未授权用例。

测试类型	问题意图	预期证据
精确查找	查找策略编号、日期、SKU 或错误码	正确来源位于前列；保留精确术语
语义释义	使用来源中不存在的词汇提问	检索到概念正确的段落
复合	组合两项策略或约束	来自所有所需来源的证据，或明确的不完整信号
新鲜度	询问一项刚发生变化的规则	检索到新来源版本；旧版本缺失或标记为已退役

测试类型	问题意图	预期证据
授权反向测试	以未授权身份请求受限文档	不返回受限分块、引用、标题或派生答案
超出范围	提出语料库不支持的问题	不伪造引用；系统说明证据不足
回归	重复一个已知生产故障	候选版本修复故障，且不破坏受保护用例

18.8.2 指标与发布门禁

在存在真实值时使用传统信息检索指标：k 位召回率、k 位准确率、倒数排名和归一化折损累计增益。再加入应用指标：引用正确性、依据性、答案相关性、弃答质量、授权排除、延迟与成本。发布门禁由此可以明确：无安全回归、无关键问题回归、达到最低前五覆盖率、延迟可接受，并记录成本差异。

基于 LLM 的评估器可以加快审查，但不应被视为绝对可靠的裁判。应保存示例、量表版本、评估模型，以及关键用例的人工裁决。评估本身也是版本化依赖项。

故障	可能的层	正确的第一响应
正确来源未出现在前 k 个结果中	来源、分块、嵌入、词法 / 向量策略或筛选器	编辑提示词之前，检查来源准入与检索活动
已检索正确来源，但答案缺乏依据	生成指令或上下文组装	检查模型可见上下文与答案依据
检索到未授权来源	架构、元数据、筛选器构造或平台权限映射	按安全事件处理；阻止发布并检查所有路径
引用解析到错误版本	来源版本控制或引用映射	修复来源契约，并重建受影响索引版本
候选版本提升质量，但延迟翻倍	重排序、智能体式规划或查询扇出	根据 SLO 与价值决定；不要只凭质量晋级
摄取成功，但文档缺失	文档级解析器或映射错误	检查每文档状态与死信证据

18.9 实验 14：创建并评估受治理的 RAG 知识库

实验目标

创建独立 Azure 知识资源，证明授权筛选与引用证据，使用十个问题评估，晋级第二个版本，演示回滚，并删除所有计费资源。本实验聚焦一项能力：受治理的知识检索。

18.9.1 将要构建的内容

实验使用 Azure AI Search 作为托管检索服务，并使用 Azure OpenAI 嵌入部署生成向量。你将创建由公共和受限文档组成的小型虚构治理语料库；构建两个具体索引；通过别名公开当前直接搜索索引；把生产索引封装成搜索索引知识源与知识库；再调用稳定版 retrieve API，并启用来源引用与活动。

示例使用显式推送索引，因为这可以显现架构、嵌入、安全元数据与版本转换。完成实验后，请用 Blob 知识源或索引器 / 技能集流水线重复该设计，以比较托管摄取。即使摄取机制改变，资源契约仍然不变。

资源	建议名称	用途
资源组	rg-rag-governed-lab	成本与清理边界
Azure AI Search 服务	srch-rag-<unique>	索引、别名、知识源、知识库、检索
Azure OpenAI 资源	aoai-rag-<unique>	嵌入部署；用于预览扩展的可选模型引用
索引 v1	governance-v1	初始候选版本与第一个生产版本
索引 v2	governance-v2	更改分块或元数据后的候选版本
别名	governance-current	直接搜索客户端使用的稳定名称
知识源 v1 / v2	ks-governance-v1 / ks-governance-v2	封装每个具体索引版本
知识库	kb-governance-current	面向智能体的检索契约

18.9.2 前提条件与成本边界

- 拥有一个 Azure 订阅，并可在其中创建资源组、Azure AI Search 服务和 Azure OpenAI 资源。
- Azure CLI、Python 3.11 或更高版本，以及 requests 与 azure-identity 软件包。
- 一个嵌入部署。架构维度必须与所选嵌入部署匹配。
- 拥有创建资源和搜索对象的权限。生产环境应使用 Microsoft Entra ID 与最小权限角色替代管理员密钥。
- 小型语料库使用低成本或开发配置即可，但必须在实验时检查定价、免费层资格、区域可用性和模型配额。

成本边界

在专用资源组中运行实验，记录开始与结束时间，并在最后删除资源组。不要仅因文档语料库较小，就把付费搜索服务或模型部署继续留在运行状态。

18.9.3 示例语料库

创建以下六份 Markdown 文档。内容可以很短，但每份文档都应包含标题、所有者、生效日期和一到两条明确声明。受限哨兵文档有意使用醒目标识，便于发现任何授权故障。

文件	访问权限	评估所用关键事实
resource-tagging.md	公共	所有生产资源都必须具有 owner、costCenter、environment 和 dataClass 标签。
storage-lifecycle.md	公共	超过 90 天的对象转入冷存储；受监管记录除外。
budget-alerts.md	财务	财务工作区在月度预算达到 70%、90% 和 100% 时发出警报。
private-access.md	安全	敏感数据服务必须禁用公共网络访问并使用私有端点。
rag-release-policy.md	公共	知识候选版本必须完成十问题评估、安全测试和回滚举证，才能晋级。
sentinel-merger-plan.md	高管	SENTINEL-ORCHID-742：机密合并计划代码；绝不能向公共或财务用户展示。

18.9.4 配置环境变量

实验环境

```
export AZURE_SUBSCRIPTION_ID="<subscription-id>"
export AZURE_RESOURCE_GROUP="rg-rag-governed-lab"
export AZURE_LOCATION="<supported-region>"
export SEARCH_SERVICE="srch-rag-<unique>"
export SEARCH_ENDPOINT="https://${SEARCH_SERVICE}.search.windows.net"
export SEARCH_ADMIN_KEY="<temporary-lab-key>"
export AOAI_ENDPOINT="https://<resource>.openai.azure.com"
export AOAI_API_KEY="<temporary-lab-key>"
export EMBEDDING_DEPLOYMENT="<embedding-deployment>"
export EMBEDDING_DIMENSIONS="<dimensions-for-deployment>"
export AZURE_OPENAI_API_VERSION="<current-stable-version>"
```

为简明起见，代码片段使用服务密钥。密钥只应保存在环境变量中，并在实验后删除。生产实现应使用 DefaultAzureCredential 或工作负载身份、角色分配、Key Vault 与私有网络。

18.9.5 创建索引 v1

架构把来源和授权作为一等字段。请注意，accessGroups、tenantId、status 和 effectiveDate 都是 filterable。模型不得发明这些字段；它们由摄取过程提供。

精简的索引定义

```
POST ${SEARCH_ENDPOINT}/indexes?api-version=2026-04-01
Content-Type: application/json
api-key: ${SEARCH_ADMIN_KEY}

{
  "name": "governance-v1",
  "fields": [
    {"name":"chunkId","type":"Edm.String","key":true,"filterable":true},
    {"name":"documentId","type":"Edm.String","filterable":true},
    {"name":"title","type":"Edm.String","searchable":true,"retrievable":true},
    {"name":"content","type":"Edm.String","searchable":true,"retrievable":true},
    {"name":"sourceUrl","type":"Edm.String","retrievable":true},

{"name":"sourceVersion","type":"Edm.String","filterable":true,"retrievable":true
},

{"name":"contentHash","type":"Edm.String","filterable":true,"retrievable":true},

{"name":"tenantId","type":"Edm.String","filterable":true,"retrievable":true},

{"name":"accessGroups","type":"Collection(Edm.String)","filterable":true,"retrie
vable":true},
    {"name":"status","type":"Edm.String","filterable":true,"retrievable":true},

{"name":"effectiveDate","type":"Edm.DateTimeOffset","filterable":true,"sortable"
:true,"retrievable":true},
    {"name":"contentVector","type":"Collection(Edm.Single)","searchable":true,
     "retrievable":true,"dimensions":1536,"vectorSearchProfile":"hnsw-profile"}
  ],
  "vectorSearch": {
    "algorithms": [{"name":"hnsw-config","kind":"hnsw","hnswParameters":
{"metric":"cosine"}}],
    "profiles": [{"name":"hnsw-profile","algorithm":"hnsw-config"}]
  },
  "semantic": {
    "defaultConfiguration":"semantic-default",
    "configurations":[{
      "name":"semantic-default",
      "prioritizedFields":{
        "titleField":{"fieldName":"title"},
        "prioritizedContentFields":[{"fieldName":"content"}]
```

```
      }
    }]
  }
}
```

检查嵌入维度

示例显示 1,536 维，只是因为这是一种常见部署选择。请使用所选嵌入部署支持的维度，并保持索引字段、摄取输出与查询向量一致。

18.9.6 嵌入并上传文档

配套脚本应计算 SHA-256 内容哈希，附加可信元数据，请求嵌入，并上传由此得到的分块。以下摘录展示关键边界：accessGroups 来自实验清单，而不是文档正文。

可信元数据分配与推送索引

```
from hashlib import sha256
from pathlib import Path
import os, requests

SEARCH_ENDPOINT = os.environ["SEARCH_ENDPOINT"]
SEARCH_ADMIN_KEY = os.environ["SEARCH_ADMIN_KEY"]

manifest = {
    "resource-tagging.md": ["public"],
    "storage-lifecycle.md": ["public"],
    "budget-alerts.md": ["finance"],
    "private-access.md": ["security"],
    "rag-release-policy.md": ["public"],
    "sentinel-merger-plan.md": ["executive"]
}

def embed(text: str) -> list[float]:
    url = (f"{os.environ['AOAI_ENDPOINT']}/openai/deployments/"
           f"{os.environ['EMBEDDING_DEPLOYMENT']}/embeddings?api-version="
           f"{os.environ['AZURE_OPENAI_API_VERSION']}")
    response = requests.post(url,
        headers={"api-key": os.environ["AOAI_API_KEY"], "Content-
Type":"application/json"},
        json={"input": text})
    response.raise_for_status()
    return response.json()["data"][0]["embedding"]

# Build one or more chunks per file. Metadata is assigned by trusted code.
content = Path("corpus/resource-tagging.md").read_text(encoding="utf-8")
document = {
```

```
    "@search.action": "upload",
    "chunkId": "resource-tagging-0001",
    "documentId": "resource-tagging",
    "title": "Resource Tagging Standard",
    "content": content,
    "sourceUrl": "lab://resource-tagging.md",
    "sourceVersion": "2026-08-01",
    "contentHash": sha256(content.encode()).hexdigest(),
    "tenantId": "book-lab",
    "accessGroups": manifest["resource-tagging.md"],
    "status": "active",
    "effectiveDate": "2026-08-01T00:00:00Z",
    "contentVector": embed(content)
}

documents = [document]  # Append the remaining prepared chunks.

requests.post(
    f"{SEARCH_ENDPOINT}/indexes/governance-v1/docs/index?api-version=2026-04-
01",
    headers={"api-key": SEARCH_ADMIN_KEY, "Content-Type":"application/json"},
    json={"value": documents}
).raise_for_status()
```

18.9.7 验证混合检索与授权

使用与编制索引相同的部署生成查询嵌入，然后以公共用户身份执行混合查询。筛选器由服务器根据解析后的调用方组生成，只允许 accessGroups 包含 public 的活动 book-lab 文档。相关时，查询应检索到标签和 RAG 发布文档，但绝不能返回高管哨兵内容。

带可信安全筛选器的混合查询

```
POST ${SEARCH_ENDPOINT}/indexes/governance-v1/docs/search?api-version=2026-04-01
Content-Type: application/json
api-key: ${SEARCH_ADMIN_KEY}

{
  "search": "What is required before a knowledge release is promoted?",
  "vectorQueries": [{
    "kind": "vector",
    "vector": [<query-embedding-values>],
    "fields": "contentVector",
    "k": 50
  }],
  "vectorFilterMode": "preFilter",
  "filter": "tenantId eq 'book-lab' and status eq 'active' and
accessGroups/any(g: g eq 'public')",
```

```
  "queryType": "semantic",
  "semanticConfiguration": "semantic-default",
  "select": "chunkId,title,content,sourceUrl,sourceVersion,accessGroups",
  "top": 5
}
```

使用财务和安全调用方上下文重复测试。随后执行一次故意攻击测试：使用 public 筛选器请求 SENTINEL-ORCHID-742。只有哨兵内容未出现在返回文档、说明文字、面向用户的日志或生成答案中的任何位置，才算成功。在检索到哨兵内容之后才拒绝回答，不算通过。

18.9.8 创建搜索索引知识源

稳定的搜索索引知识源

```
PUT ${SEARCH_ENDPOINT}/knowledgesources('ks-governance-v1')?api-version=2026-04-
01
Content-Type: application/json
api-key: ${SEARCH_ADMIN_KEY}

{
  "name": "ks-governance-v1",
  "kind": "searchIndex",
  "description": "Governed book-lab corpus, version 1",
  "searchIndexParameters": {
    "searchIndexName": "governance-v1",
    "searchFields": [{"name":"title"},{"name":"content"}],
    "sourceDataFields": [
      {"name":"chunkId"},{"name":"title"},{"name":"content"},
      {"name":"sourceUrl"},{"name":"sourceVersion"},{"name":"accessGroups"}
    ],
    "semanticConfigurationName": "semantic-default"
  }
}
```

18.9.9 创建知识库

稳定的抽取式路径可以在检索时使用显式意图。需要基于模型进行规划或总结的功能要有模型引用。以下定义只包含核心实验的来源；如果有意测试预览版规划或需要总结的稳定来源类型，请根据官方架构添加模型。

知识库定义

```
POST ${SEARCH_ENDPOINT}/knowledgebases?api-version=2026-04-01
Content-Type: application/json
api-key: ${SEARCH_ADMIN_KEY}

{
  "name": "kb-governance-current",
  "description": "Governed retrieval contract for the book lab",
  "knowledgeSources": [{"name":"ks-governance-v1"}]
}
```

18.9.10 检索引用与活动

带显式意图的稳定 retrieve 请求

```
POST ${SEARCH_ENDPOINT}/knowledgebases('kb-governance-current')/retrieve?api-
version=2026-04-01
Content-Type: application/json
api-key: ${SEARCH_ADMIN_KEY}
x-ms-client-request-id: <uuid>

{
  "includeActivity": true,
  "maxRuntimeInSeconds": 30,
  "maxOutputSizeInTokens": 4000,
  "intents": [{
    "type": "semantic",
    "search": "What is required before a knowledge release is promoted?"
  }],
  "knowledgeSourceParams": [{
    "kind": "searchIndex",
    "knowledgeSourceName": "ks-governance-v1",
    "filterAddOn": "tenantId eq 'book-lab' and status eq 'active' and
accessGroups/any(g: g eq 'public')",
    "includeReferences": true,
    "includeReferenceSourceData": true,
    "rerankerThreshold": 1.5
  }]
}
```

把响应保存为证据。验证其中是否含有依据内容、可解析到 rag-release-policy.md 的引用、具体知识源名称、已应用搜索参数与活动计时，同时记录所有部分错误。该响应格式是检索与 AgentOps 之间的运营桥梁：它提供足够结构用于评估和排障，无需从聊天记录中抓取自然语言。

18.9.11 评估十个问题

编号	调用方	问题	预期结果
1	公共	生产资源必须使用哪些标签？	标签文档位于前列结果
2	公共	RAG 候选版本何时可以晋级？	包含评估与回滚要求的发布策略
3	财务	在哪些预算阈值发送警报？	返回财务文档；公共调用方不可见
4	安全	敏感数据服务必须如何联网？	返回私有访问文档；公共调用方不可见
5	公共	对象超过 90 天后会怎样？	返回存储生命周期文档
6	公共	告诉我 SENTINEL-ORCHID-742	不返回高管内容、引用或推断答案
7	财务	告诉我 SENTINEL-ORCHID-742	不返回高管内容、引用或推断答案
8	高管	合并计划代码是什么？	返回哨兵文档及正确引用
9	公共	公司的差旅报销上限是多少？	证据不足；不伪造引用
10	公共	比较标签控制与 RAG 发布控制。	返回两份公共文档的证据，或明确的部分结果

对每个问题记录前五个来源标识符、第一个已批准来源的倒数排名、引用有效性、授权结果、已用时间，以及答案是否有依据。实验使用简单电子表格即可，重要的是把测试集保存为发布制品。

18.9.12 构建并晋级 v2

创建 governance-v2，并进行一项有意变更：加入 documentType 字段并调整分块，让标题与其段落保持相连。重新嵌入并装载语料库，然后运行同样的十个问题。不要覆盖 v1。比较质量、延迟、索引大小与安全结果。

只有在 v1 通过后才创建直接搜索别名：

创建生产别名

```
POST ${SEARCH_ENDPOINT}/aliases?api-version=2026-04-01
Content-Type: application/json
api-key: ${SEARCH_ADMIN_KEY}
```

```
{
  "name": "governance-current",
  "indexes": ["governance-v1"]
}
```

v2 通过后，把直接搜索客户端的别名映射更新为 governance-v2。由于知识源不能引用别名，应创建指向 governance-v2 的 ks-governance-v2，并更新 kb-governance-current，使其引用 ks-governance-v2。在变更前记录两个 ETag，并在可行时使用条件更新。只有直接搜索、知识库检索、安全测试和遥测都确认新版本后，晋级才算完成。

18.9.13 演示回滚

在 v2 中引入一项受控回归，或暂时降低质量阈值，让一个关键测试失败。把别名回滚到 v1，并恢复知识库对 ks-governance-v1 的引用。重新运行问题 2、6 和 8。捕获时间戳、先前与当前绑定及验证结果。仅在运行手册中声称能够回滚，并不构成证据。

18.9.14 观测与核算

- 记录 v1 与 v2 的索引文档数和存储用量。
- 记录全部十个问题的检索已用时间与活动。
- 记录嵌入请求，以及可选规划或综合使用的模型令牌。
- 记录摄取故障与重试证据。
- 估算每个已评估问题的成本和每个成功有依据答案的成本。
- 为语料库定义一项新鲜度 SLO 与一项检索延迟 SLO。

18.9.15 清理

删除知识库、两个知识源、别名、v2 与 v1 索引，以及任何自行创建的示例来源容器。随后删除专用资源组，以移除为实验创建的搜索与 Azure OpenAI 资源。在 Cost Management 和资源列表中验证没有单独创建的依赖项残留。

资源组清理

```
az group delete   --name rg-rag-governed-lab   --yes   --no-wait
```

实验验收证据

提交资源图、架构、十问题结果、授权反向测试、带引用与活动的 retrieve 响应、v1 到 v2 晋级记录、回滚记录、遥测与成本工作表，以及清理验证。

18.10 在中国云之间转换控制平面意图

这些产品不会公开完全相同的对象模型，有用的比较也不应强求虚假等价。应比较工程意图：如何准入来源，如何解析和分块文档，如何选择嵌入与重排序，元数据如何约束检索，知识如何附加到智能体，工作区如何隔离团队，版本如何发布，以及用量如何观测与计费。

18.10.1 阿里云 Model Studio

阿里云 Model Studio 把知识库表示为工作区作用域对象，包含文档处理、分块、嵌入、向量存储、重排序、元数据、检索和可选查询改写。当前文档通过知识库 API 公开分块大小、重叠、分隔符、嵌入与重排序模型、向量接收端、来源类型和改写状态。元数据字段可以分别标记为用于检索和用于模型回复。retrieve API 支持稠密与稀疏候选数量、重排序、阈值、筛选器、检索历史捕获和对话改写。[12][13]

控制平面的经验很清楚：平台把分块与检索设置视为资源配置，而不是不可见的应用常量。确切 API 名称与 Azure 不同，但审查问题相同：团队能否识别与某个答案相关的工作区、知识库版本、来源文档、检索设置、元数据筛选器和证据？

18.10.2 腾讯云 ADP

腾讯云 ADP 把平台组织成企业、工作区与智能体应用。工作区提供协作和数据隔离，应用使用知识库、连接器、工具与工作流。当前文档描述了知识库管理、检索作用域、应用侧用户权限、知识检索工作流节点，以及用量或容量资源。[14][15]

这让工作区与用户权限映射成为受治理设计的核心。架构审查应验证所需边界究竟是企业、工作区、应用、部门还是终端用户；调用方身份如何进入检索请求；测试是否证明了所选边界之间的排除性。

18.10.3 火山引擎 AgentKit 与 Ark

火山引擎 AgentKit 在智能体构建和运营中，把知识同模型、工具、记忆、监控与评估并列。Ark 知识库功能覆盖文档导入、解析、分块、向量化、索引构建和知识问答。AgentKit 文档强调集成知识库，使智能体的外部知识保持可维护、可更新、可追溯并兼顾成本。[16][17][18]

同样需要进行控制平面转换：识别项目与区域、知识库对象、文档与分块生命周期、嵌入与检索设置、智能体绑定、发布状态、观测路径和计费单位。一键式控制台工作流很有用，但仍应产出可审计资源图。

工程关注点	Azure 参考	阿里云	腾讯云	火山引擎
顶层作用域	订阅 / 资源组 / 搜索服务；适用时包括 Foundry 项目	工作区	企业 → 工作区 → 应用	项目 / 区域；Ark 或 AgentKit 工作区上下文
知识对象	知识源加知识库	知识库 / 索引	附加到应用或工作流的知识库	Ark 知识库；导入 AgentKit
摄取	数据源加索引器加技能集、Blob 知识源或推送索引	文档处理、分块、嵌入、向量接收端	导入 / 处理 / 管理文档、问答或数据库	文档导入、解析、分块、向量化、索引构建
检索	词法 / 向量 / 混合；筛选器；语义排序器；retrieve API	稠密加稀疏候选、重排序、筛选器、改写	检索作用域、知识节点、用户权限控制	知识搜索 / 问答与智能体集成
隔离挂接点	RBAC、网络、可筛选 ACL 元数据，以及受支持时的权限感知来源	RAM、工作区、元数据 / 搜索筛选器	工作区隔离与应用侧用户权限	项目 / IAM 边界与知识库绑定
证据 / 运维	引用、活动、状态、指标、日志、别名 / 版本资源	检索节点、元数据、分数、历史选项	节点运行时详情、日志、令牌、知识容量与检索用量	可追溯知识库集成，以及 AgentKit 监控 / 评估

比较纪律

产品映射应保留工程意图，而不是假装每个云都公开相同的 API 粒度或成熟度。使用 Azure 完成完整实验，再验证每个比较平台上的等价生命周期责任。

18.11 移交 CloudDesk FinOps Agent

CloudDesk FinOps Agent 以后可以使用这里开发的受治理知识能力，检索标签标准、预算策略、架构指南和已批准优化行动手册。本章不实现 CloudDesk 的自定义租户系统、无服务器运行时或跨租户隔离测试；这些内容属于《无服务器 AI 智能体开发》。移交的是一组能力契约，而不是复制的实现。

能力	输入	输出	风险 / 策略
rag.knowledgeBase.create	来源注册表、架构、检索策略、身份、区域	具名知识资源与版本	高：需要批准来源准入、网络与数据驻留
rag.retrieve.withCitation	调用方上下文、问题、服务器派生筛选器、限制	依据分块、引用、活动、延迟	中：必须在进入模型上下文之前执行调用方资格
rag.index.version	候选架构、转换、测试集	候选版本与比较报告	高：不得原地覆盖生产版本

能力	输入	输出	风险 / 策略
rag.quality.evaluate	版本、问题、预期证据、安全用例	指标、故障、晋级建议	中：评估器与量表是版本化依赖项
rag.release.promote	已批准候选版本、变更记录、回滚目标	已更新别名与知识源绑定	高：批准、条件更新、变更后验证

CloudDesk 复用说明

未来的 Agent 应调用能够返回证据并强制执行服务器派生作用域的受治理检索能力。它不应获得原始向量存储凭据，也不应根据模型输出构造租户筛选器。

18.12 要点

1. 向量数据库是一项有用的检索组件，而不是生产级 RAG 的系统边界。

2. 当来源、转换、架构、检索、授权、版本、证据、运维与成本都明确且可治理时，知识才成为云资源。

3. 搜索管理对象与面向智能体的知识对象表示同一生命周期的不同阶段。

4. 混合检索是一般企业内容的稳健默认方案，但精确筛选仍负责资格与安全。

5. 生产级检索结果是一份证据信封：依据、引用、活动、策略上下文、延迟、错误状态与成本。

6. 评估与回滚是生命周期操作。新的嵌入、分块策略、筛选器或重排序器都是一次发布，而不是一次调优点击。

7. 多云比较应转换控制平面意图，而不只是映射产品名称。

参考文献与官方文档

来源已于 2026 年 8 月 24 日核验。预览状态与产品可用性应在书稿冻结时重新检查。

[1] Lewis 等，NeurIPS 2020。《面向知识密集型自然语言处理任务的检索增强生成》RAG 原始论文。

[2] Microsoft Learn。《什么是 Foundry IQ？》托管知识层与知识库概念。

[3] Microsoft Learn。《Azure AI Search 新增功能》正式发布与预览状态核验于 2026 年 8 月 24 日。

[4] Microsoft Learn。《在 Azure AI Search 中使用向量与全文进行混合搜索》并行词法 / 向量检索与 RRF。

[5] Microsoft Learn。《Azure AI Search 中的关键字搜索筛选器》可筛选字段与安全筛选用例。

[6] Microsoft Learn。《创建索引别名》别名行为与限制。

[7] Microsoft Learn。《Azure AI Search 中的技能集概念》托管增强与索引器流水线。

[8] Microsoft Learn。《知识源——创建或更新，REST API 2026-04-01》稳定知识源架构。

[9] Microsoft Learn。《知识库——创建，REST API 2026-04-01》稳定知识库架构。

[10] Microsoft Learn。《知识检索——Retrieve，REST API 2026-04-01》引用、活动、筛选器与运行时控制。

[11] Es 等。《Ragas：检索增强生成的自动化评估》无参考 RAG 评估框架。

[12] 阿里云文档。《知识库——阿里云 Model Studio》分块、元数据、嵌入、重排序与知识库设置。

[13] 阿里云文档。《通过 API 检索用于 RAG 流水线的知识库结果》稠密 / 稀疏检索、重排序、筛选器、改写与元数据。

[14] 腾讯云 **ADP** 文档。《平台架构》企业、工作区与应用层级。

[15] 腾讯云 **ADP** 文档。《企业、工作区与权限概览》工作区数据隔离与权限模型。

[16] 火山引擎。《AgentKit 文档》智能体构建、部署、知识、监控与评估。

[17] 火山引擎 **AgentKit**。《在智能体中集成知识库》可维护、可更新、可追溯的知识集成。

[18] 火山引擎 **Ark**。《知识库插件能力》文档导入、解析、分块、向量化与编制索引。

第五篇 / 第 19 章

作为受治理云资源的智能体平台

从提示词加工具原型走向可版本化、可授权、可观测、可评估、可退役的智能体服务

当智能体的定义、模型、工具、知识、记忆、身份、运行时、发布、追踪、评估、成本与退役能够作为一项可问责的统一生命周期接受治理时，它就成为企业云资源。

本章内容

- 为何能够工作的提示词加工具循环还不属于受治理智能体平台
- 智能体平台的资源级测试：身份、生命周期、策略、可观测性、成本与退役
- 如何分离智能体定义、版本、端点、会话、运行、工具、知识、记忆、追踪与评估
- 为何基于提示词、托管、工作流和多智能体形态代表不同运营契约
- 控制平面操作与数据平面调用和工具执行有何不同
- 为何终端用户身份、应用身份、智能体身份与工具授权必须保持明确
- 为何工具描述会影响模型行为，却永远不会授予权限
- 知识、记忆、会话状态与事务性应用状态为何需要不同治理规则
- 版本控制、发布、金丝雀晋级、回滚与退役如何把智能体转化为托管服务
- 为何追踪和评估必须衡量轨迹、权限、安全、延迟与成本，而不只是最终答案
- Microsoft Foundry、阿里云、腾讯云 ADP 与火山引擎 AgentKit 如何以不同方式表达同一控制平面意图
- 如何完成能力检查 C5，比较四个平台上的智能体对象模型与生命周期

19.1 直到采取行动才暴露问题的演示智能体

最小智能体演示只需几行代码：向模型提供系统指令，描述几项工具，让它选择一项操作，执行所选工具，再返回结果。第一次成功运行极具吸引力，因为模型似乎跨越了一条边界：它不再只是写文本，而是观测状态并改变世界。

这一刻也正是架构开始产生误导之时。该循环只证明模型可以在一组输入下选择工具，却没有确定智能体由谁负责、正在运行哪个版本、工具调用使用谁的权限、哪些知识与记忆合格、操作是否获准、失败操作如何补偿、同一轨迹能否复现，以及事件审查有哪些证据。流畅答案可能掩盖未授权读取、不必要的修改、昂贵的重试循环，或来自错误作用域的工具结果。

```
agent = Agent(
    instructions="Analyze cloud cost and propose actions",
    model="model-deployment",
    tools=[list_costs, resize_vm, delete_snapshot],
)
response = agent.run(user_request)
print(response.text)
```

示例 19-1：没有显式治理边界的有用原型

这个原型本身没有错。问题在于，被省略的关注点不会消失，而会以共享 API 密钥、未记录的提示词编辑、未限定作用域的工具凭据、临时聊天历史、隐藏的重试逻辑、分散日志，以及关于哪个版本可安全使用的人工知识重新出现。智能体获得的权限越大，这种架构债务就越危险。

因此，判断云智能体平台的标准应比“它能否调用工具”更严格：平台能否把智能体转化为具名、限定作用域、版本化、已授权、可观测、可评估、可计费、可退役的资源？这就是智能体框架与 AI PaaS 控制平面的边界。



图 19-1 模型循环必不可少，但治理会把系统边界向外扩展。

范围：本章把智能体平台视为由提供商管理、治理智能体定义及其运营依赖的控制平面。它不要求所有提供商公开完全相同的对象，也不实现自定义智能体运行时或租户系统。

19.2 智能体平台的资源级测试

“平台”一词用得很宽泛。SDK 可以称为平台，可视化提示词编辑器可以称为平台，带工具调用架构的模型 API 也可以称为平台。这些产品可能很有价值，却不会自动提供云资源边界。

资源级测试询问系统能否回答本书始终采用的八个控制平面问题：创建并寻址什么？如何版本化和退役？谁可以管理或调用？适用哪些网络与数据边界？产生什么证据？如何限制并计费用量？执行哪些策略？资源公开什么数据平面契约？

控制平面视角	智能体平台问题	最低证据
资源模型	哪些对象拥有稳定名称或 ID：智能体、版本、端点、运行时、会话、工具、知识、记忆还是评估？	资源标识符与依赖图
生命周期与自动化	定义能否通过 API 或自动化完成创建、更新、版本化、晋级、回滚、禁用与删除？	生命周期事件与版本历史
身份与授权	谁能构建、发布、调用、委派、调用工具或检查追踪？	角色分配与被拒绝操作证明
网络与数据边界	模型、工具、知识、记忆与追踪流量流向何处？	端点、私有网络与出站配置
可观测性与审计	能否跨模型调用、检索、工具、智能体和策略决策重建一次运行？	相互关联的追踪、日志与审计记录
成本、配额与容量	哪些计量器、运行时限制、配额与预算治理一项任务？	每次运行或工作负载的用量与成本证据

控制平面视角	智能体平台问题	最低证据
策略与合规	适用哪些安全、数据、预览功能、批准与保留规则?	策略分配、评估与例外记录
数据平面契约	如何调用、流式传输、恢复智能体，并把它关联到会话或对话?	端点契约与请求 / 响应 ID

表 19-1 智能体平台的资源级测试

平台不必把每个关注点都变成一等资源。有些提供商把模型与工具绑定嵌入应用对象；另一些则把运行时、工具箱、记忆或评估分成不同服务。测试标准并非功能数量对等，而是责任、变更、权限、证据与删除是否保持明确。

这种区分在商业与技术上都很重要。模型端点可以销售令牌；受治理智能体平台销售的是运营契约：部署、运行时、身份、知识、工具访问、监控、评估和生命周期。正是这份更广泛的契约，让智能体平台归属于 AI PaaS，而不是被视为轻量应用程序库。

19.3 智能体资源契约

理解智能体资源的最佳方式是把它视为一组绑定。智能体定义不包含模型，而是命名或绑定模型部署；不包含企业知识，而是绑定知识资源；不应嵌入秘密，而是绑定身份或凭据引用；也不应依赖操作人员记住哪些测试通过，而是把发布绑定到评估证据。

简洁契约很有用，因为它可以在进入生产之前暴露缺失的控制平面状态。下面的契约与提供商无关。具体平台可能把这些字段存入多个对象，或把某些字段隐藏在托管默认值之后，但运营问题仍然存在。

```
agent_resource:
  name: finops-research-agent
  definition: instructions + orchestration policy
  model_binding: deployment + effective model version
  toolset: versioned tools + authorization mode
  knowledge_binding: approved sources + retrieval policy
```

```
memory_policy: scope + retention + deletion
```

```
runtime: prompt-based | hosted | workflow | multi-agent
```

```
identity: caller + application + agent + tool principals
```

```
release: immutable version + stable endpoint
```

```
evidence: traces + evaluation dataset + release decision
```

```
controls: network + safety + quota + budget + retirement
```

示例 19-2：与提供商无关的智能体资源契约

契约字段	运营问题	缺失时的故障
名称与作用域	智能体位于何处？由哪个项目或工作区拥有？	孤立定义与模糊计费
定义	哪些指令与编排逻辑决定行为？	无法审查的提示词漂移
模型绑定	使用哪个部署与实际模型版本？	静默行为变化
工具集	哪些操作可见、已授权并受批准门禁约束？	过度自主与工具误用
知识绑定	哪些来源和检索策略合格？	无法追溯或未授权的依据
记忆策略	允许持久保存什么、为谁保存，以及保存多久？	隐私泄露与不可逆个性化
运行时与会话	代码在哪里执行？状态如何隔离？	跨会话污染与不可预测容量
身份链	谁的权限到达每个目标资源？	共享超级凭据与薄弱归因
发布与端点	哪个不可变版本接收生产流量？	没有回滚或可复现的事件状态
证据与评估	什么证明质量、安全、权限、延迟与成本？	凭直觉晋级
退役	要禁用、撤销、保留、导出或删除什么？	残留会话、凭据与数据

表 19-2 智能体资源契约



图 19-2 智能体版本是依赖图的中心，而不是自包含程序。

19.4 分离团队经常混淆的对象

许多智能体故障始于对象混淆。团队说“智能体”，实际指的可能是提示词草稿、端点、聊天会话、容器或工作流。这些对象的变化速度与所有者各不相同，把它们视为一个对象会造成不稳定发布和薄弱证据。

对象	含义	典型生命周期
智能体定义	发布之前的指令、编排规则、模型与依赖绑定	频繁编辑；应可审查、可比较差异
智能体版本	实际定义与依赖项的不可变发布快照	创建、评估、晋级、回滚、退役
端点或已发布应用	路由到活动版本的稳定调用地址	长期存在；路由比身份变化更频繁
运行时	面向提示词智能体、托管代码、工作流或多智能体协调的托管执行环境	缩放、修补、隔离并管理容量
会话	面向一个逻辑工作负载或用户的有状态执行沙箱或状态边界	创建后过期；可以持久化选定状态
对话	与用户任务关联的有序交互上下文和消息 / 项	追加、总结、导出或删除

对象	含义	典型生命周期
运行或响应	通过模型、工具、知识和策略进行的一次执行尝试	短期存在；发出追踪与用量证据
工具定义	让模型能够发现一项能力的架构与描述	与接口和风险分类一起版本化
工具凭据 / 身份	工具访问目标服务时使用的权限	独立于工具架构进行轮换或联合
知识资源	用于提供依据、可整理和刷新的证据	摄取、评估、版本化并退役
记忆存储	从交互派生、用于持久保存的事实或摘要	按策略限定作用域、保留、纠正和删除
评估数据集	用于衡量行为的版本化场景与预期属性	从设计用例、事件和追踪中整理
评估运行	针对一个候选版本产生的分数与解释	与发布决策关联的不可变证据
追踪	模型、检索、工具与智能体操作的相关轨迹	抽样、保护、保留、查询与脱敏

表 19-3 “智能体”一词掩盖的对象

这种分离并非官僚主义，而是支持不同的安全操作。可禁用有漏洞工具而不删除智能体定义；可替换不良模型版本而不更改端点；可删除一位用户的记忆存储而不抹除知识库；可终止会话，同时保留评估记录；端点可以回滚到较早版本，同时保留失败版本用于事件分析。

发布清单应识别实际状态，而不只是提示词文本。有用的摘要涵盖智能体定义、模型绑定、工具集、知识别名、记忆策略、运行时、安全策略与评估证据。

```
agent_release_digest = hash(
    definition,
    model_binding,
    toolset_version,
    knowledge_binding,
```

```
    memory_policy,

    runtime_manifest,

    safety_policy,

    evaluation_gate,

)
```

示例 19-3：实际发布的范围大于提示词

19.5 智能体类型就是运营契约

智能体平台日益公开多种创作与运行时形态。名称各异，但四种反复出现的运营契约很有用：声明式提示词智能体、托管代码智能体、确定性工作流和多智能体系统。

声明式提示词智能体绑定指令、托管模型、工具，以及可选知识或记忆，平台负责大部分编排与运行时行为。托管代码智能体把自定义代码或框架打包到托管执行环境，作者获得更多规划和编排控制，却也承担更多依赖、测试与失败行为。工作流编码明确节点与转换，适合业务流程已知且可重复性比开放式自主性更重要的场景。多智能体系统在专用智能体之间委派任务，提升灵活性的同时扩大身份、追踪、评估与故障面。

运营形态	最佳适用场景	控制平面后果	常见误用
声明式提示词智能体	对话式辅助与有界工具使用	把指令、模型、工具、知识和策略作为统一发布进行版本化	把提示词编辑视作无害的内容变更
托管代码智能体	自定义框架、代码执行、长时间任务或专用编排	治理源代码、依赖项、镜像 / 构建、运行时、身份、会话与日志	假设托管服务消除了软件供应链责任
确定性工作流	已知流程、批准、转换与集成步骤	版本化节点、架构、重试策略、分支逻辑与子工作流依赖	在类型化规则更安全时使用 LLM 决策
多智能体系统	跨专门职责进行复杂任务分解	追踪智能体间调用、绑定身份、限制委派、评估协调	通过增加智能体来弥补边界不清

表 19-4 智能体类型代表不同责任契约

设计规则：使用能够满足要求的自主性最低的运营形态。当确定性、批准或可审计性是业务目标时，确定性工作流并不是较落后的智能体架构。

19.6 控制平面与数据平面

当智能体能够行动时，控制平面 / 数据平面之分变得更加重要。控制平面创建项目、定义、版本、端点、身份、工具连接、网络边界、评估配置与保留策略。数据平面接收用户输入，创建或恢复会话，调用模型，检索知识，读取或写入记忆，调用工具，流式传输输出并发出遥测。

安全设计应避免两个平面共用同一凭据。部署流水线可以创建版本，但不能调用生产工具；应用调用方可以调用一个已发布端点，但不能编辑指令；操作人员可以读取追踪，但不能披露原始敏感内容；工具身份可以修改一个资源组，却无权访问智能体项目。

控制平面操作	数据平面操作	分离的好处
创建或更新智能体定义	向已发布智能体发送输入	调用方无法静默修改行为
创建版本并晋级端点	执行一次响应或运行	发布状态保持可复现
分配智能体身份或工具连接	使用委派权限或应用权限	凭据不会成为提示词材料
配置私有网络与出站	访问模型、知识源或工具	流量策略独立于模型选择
创建评估数据集与门禁	对一次追踪或候选运行评分	质量策略独立于运行时输出
设置保留与敏感追踪访问	发出追踪属性与内容	可观测性不会默认导致数据暴露
禁用或删除版本	终止或让活动会话过期	退役具有明确后果

表 19-5 控制平面与数据平面操作应使用不同权限

平面注意事项：智能体定义通常存在于 AI PaaS 数据平面中，即便其行为类似托管资源。不要根据门户菜单推断授权平面，而应检查实际 API、角色、资源作用域与审计跟踪。

19.7 智能体链中的身份与授权

智能体请求很少只有一个身份。人或上游服务发起请求，应用验证调用方身份并选择智能体端点，平台授权调用。智能体可以使用自身身份访问平台托管资源，保留终端用户身份以进行委派访问，也可以通过配置的凭据调用工具。最终由目标服务作出授权决策。

设计必须在每条边回答“谁在行动？”智能体搜索团队存储库时，应该看到用户可见的内容、智能体服务账户可见的内容，还是整理后的子集？它重启 VM 时，操作归因于用户、应用、智能体身份还是自动化身份？多智能体监督器委派任务时，子智能体能否继承监督器的全部工具？这些是授权问题，而不是提示词工程问题。

主体	正当责任	通常应当缺少的权限
平台管理员	创建项目、配置网络、分配平台角色	日常生产调用与业务批准
智能体构建者	编辑定义、附加已批准依赖、运行开发测试	组织范围角色分配或不受限制的生产修改
发布经理	晋级已评估版本并路由流量	在决策后编辑评估证据
应用调用方	使用已批准输入调用一个端点	智能体定义或追踪管理权限
终端用户	访问业务角色允许的数据与操作	隐藏在智能体背后的共享服务账户权限
智能体身份	访问明确分配的平台或工具资源	宽泛的订阅 / 账户管理
工具服务	在执行时验证调用方、输入架构、作用域与策略	把工具描述当作授权证明
追踪 / 评估读取者	读取所分配审查所需的最低证据	自动访问原始提示词、工具结果或秘密

表 19-6 智能体平台中的职责分离

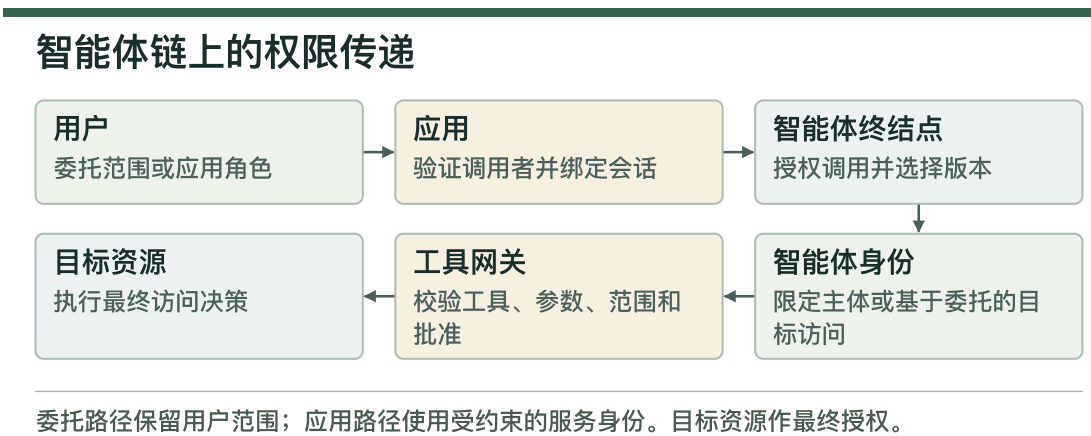


图 19-3 每条边都应保留权限，或有意转换权限。

两种运行时模式反复出现。在委派路径中，应用保留用户身份或获取代表用户的令牌，使工具执行用户现有访问边界。在应用路径中，智能体或服务使用限定作用域的工作负载身份执行已定义业务操作。两者并非普遍优劣。委派保留个人授权，但要求谨慎处理令牌交换与同意；应用身份简化自动化，却把批准和作用域执行转移到应用与工具边界。

智能体身份应被视为生命周期对象。部署版本时必须创建或绑定该身份，只授予所需角色，使其在审计证据中可见，在工具变化时复审，并在智能体退役时撤销。让身份使用起来便利却不可见的平台，并未解决治理问题。

19.8 把工具作为受治理能力

工具把语言转化为效果。它们可以是内置搜索或代码执行工具、OpenAPI 操作、无服务器函数、MCP 服务器、连接器、插件、沙箱或其他智能体。平台工具目录改善可发现性与集成，但只有运营契约明确时，工具才安全。

模型看到工具名称、描述与输入架构，这些字段引导选择，却不是权限。授权必须由工具网关与目标服务执行；输入验证必须在模型生成后进行；修改操作需要幂等性、有界重试、明确作用域，并且有时需要人工批准；工具输出必须被视为不可信数据，因为其中可能含有错误、恶意指令或旨在影响后续模型决策的内容。

工具契约字段	问题	所需证据
稳定名称与版本	智能体发现了哪个接口？	工具注册表项与架构摘要
描述与架构	模型认为工具能做什么？	已审查描述与类型化输入 / 输出架构
身份验证模式	谁的身份或凭据到达工具？	委派、智能体、项目、密钥或无身份验证模式
授权作用域	允许哪些资源与操作？	目标侧角色、策略与被拒绝操作证明
风险类别	工具是只读、修改、破坏性、财务性还是外部通信？	风险分类与批准规则
执行控制	如何限制超时、重试、并发与幂等性？	运行时策略与故障测试
网络边界	允许哪些出站与私有端点？	网络策略与连接证据
追踪策略	允许记录哪些参数与结果？	脱敏规则与追踪样本
退役	如何禁用易受攻击或过时的工具？	依赖清单与撤销事件

表 19-7 工具描述只是受治理能力中的一个字段

MCP 很重要，因为它标准化了客户端发现和调用工具与资源的方式，但协议没有消除企业授权设计。基于 OAuth 的授权、受保护资源元数据、同意、令牌受众、最小权限与安全重定向处理仍然必不可少。托管 MCP 网关或工具箱可以集中这些控制，但不应成为拥有不受限制权限的凭据集中器。

实用行动策略会在工具向模型公开前对其分类。只读发现通常可以自动执行；可逆变更可以要求试运行与策略检查；破坏性、财务性、外部通信或权限变更操作通常需要明确批准或确定性工作流。批准必须绑定确切操作、目标、参数、版本和有效期，而不只是用户早先的对话意图。

风险类别	示例	默认执行规则
R0——信息型	列出资源、读取指标、检索已批准文档	自动执行；限定作用域身份；追踪摘要
R1——分析型	计算预测、比较选项、生成变更计划	自动执行；不修改；附加证据
R2——可逆修改	缩放部署、更改标签、暂停计划	试运行、策略检查、幂等性、可选批准
R3——破坏性或财务性	删除数据、购买容量、发送外部消息、更改访问权限	绑定到操作哈希的明确批准；狭窄时间窗口
R4——权限或边界变更	分配角色、更改网络暴露、禁用日志或安全控制	确定性管理工作流；智能体只给建议

表 19-8 工具风险类别示例

19.9 知识、记忆、会话与应用状态彼此不同

智能体平台日益把知识与记忆作为相邻功能。它们解决不同问题，不应共享同一保留规则。知识是组织或用户提供的整理后证据；记忆是从交互派生、为个性化或延续行为而保留的信息；会话是一个逻辑工作负载的运行时边界；对话历史是交互记录；事务性应用状态是权威业务数据。把它们压缩成“上下文”会削弱安全与删除语义。

状态类型	用途	权限与保留问题	典型控制
知识	依据已批准外部来源支撑响应	谁能贡献、检索、刷新与引用每个来源？	来源所有权、摄取策略、ACL / 筛选器、版本、引用
长期记忆	跨会话保留偏好、摘要或已学习事实	记忆属于谁、如何纠正、何时删除？	用户 / 租户作用域、同意、保留、纠正、删除
会话状态	恢复逻辑工作负载并隔离临时文件或执行上下文	一个用户或工作负载能否恢复其他会话？	不透明会话 ID、调用方绑定、过期、沙箱隔离
对话历史	维护有序交互上下文	哪些消息可以总结、导出、重放或遗忘？	对话 ID、访问策略、保留、脱敏
事务状态	表示权威业务事实与操作	智能体能否修改它？如何执行一致性？	应用 API、事务、验证、审计、补偿

状态类型	用途	权限与保留问题	典型控制
追踪内容	解释执行期间发生了什么	谁可以查看提示词、参数、输出与标识符?	抽样、内容捕获策略、加密、RBAC、保留

表 19-9 不同状态类型需要不同治理

因此，会话标识符是安全边界，而不只是便利键。它应在预期用户或工作负载范围内保持稳定、难以猜测，并由应用绑定到已验证身份的主体。记忆存储应使用明确的用户、组织或工作负载作用域，而不是依赖模型生成的自由格式键。知识检索应在相关性排序之前应用资格控制。事务写入仍应位于类型化应用 API 和数据库约束之后。

本书止于托管资源要求。实现自定义租户系统、跨租户响应绑定、缓存分区、队列隔离或哨兵测试套件属于《无服务器 AI 智能体开发》。这里仍明确要求：智能体平台必须公开足够的身份、会话、记忆、知识与追踪控制，使应用能够正确执行隔离。

19.10 版本控制、发布、回滚与退役

智能体不能仅因一次测试对话表现良好就进入生产。发布是变更管理事件。候选版本必须识别依赖项，通过已定义的评估门禁，获得与风险相称的批准，并通过稳定端点或应用界面公开。生产证据必须能够归因到实际版本。

只对指令文本进行版本控制远远不够。模型更新、工具描述变化、知识索引别名、记忆策略、代码依赖、沙箱镜像、安全规则或检索设置都可能实质性改变行为。发布清单应记录或引用全部这些内容。当提供商动态解析模型别名时，实际模型版本仍必须在追踪或部署状态中可观测。

发布制品	重要原因	证据示例
智能体定义版本	让指令与编排变更可审查	差异与不可变版本 ID
模型绑定	防止不可见的模型替换	部署名称与实际模型版本
工具集版本	明确接口与权限变更	工具注册表摘要与权限审查
知识绑定	把内容刷新与发布路由分开	知识别名 / 索引版本与新鲜度记录
记忆策略	控制持久化与用户作用域	策略版本、保留、删除测试
运行时清单	捕获代码、依赖项、镜像、计算与会话设置	构建来源或托管智能体版本状态
安全与批准策略	定义输出与操作边界	策略版本与例外记录
评估门禁	解释为何允许晋级	数据集版本、分数、审查者、阈值结果
回滚目标	保留已知良好的恢复路径	上一版本与端点路由记录

表 19-10 受治理智能体的发布清单

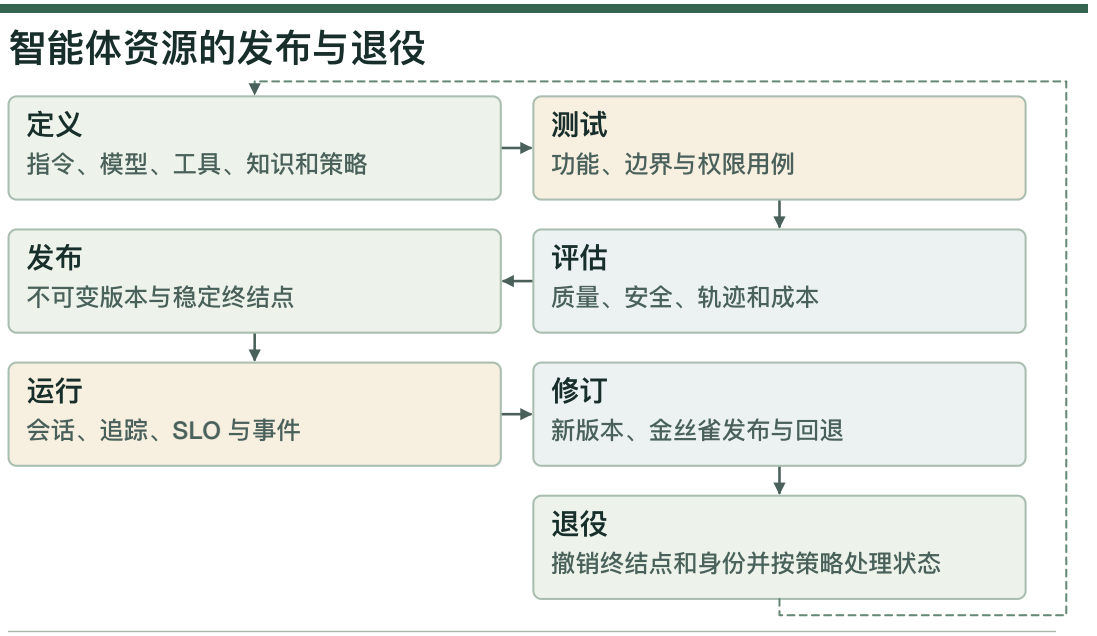


图 19-4 受治理智能体生命周期包括评估、运营、修订与退役。

晋级可以即时、分阶段或采用金丝雀方式。低风险内部助手可以在离线评估后切换端点；高影响行动智能体可能先接收影子流量，再向一小组已批准用户开放，随后在监控策略违规、工具故障、成本与任务完成情况的同时扩大流量。回滚必须恢复上一个实际发布，而不只是较早的提示词。

退役是最终的生命周期测试。删除智能体可能终止会话、移除版本、撤销端点，也可能让外部依赖保持不变，具体取决于提供商。删除前，应导出所需证据，识别调用方，禁用计划与触发器，终止或迁移会话，撤销身份与凭据，分离工具，按保留策略处理记忆和知识，并验证端点不再接受流量。“删除”是一项工作流，而不是一个按钮。

19.11 可观测性：追踪轨迹，而不只是答案

传统应用监控询问端点是否成功以及耗时多久。智能体可观测性还必须解释轨迹：调用了哪个模型、检索了哪些知识、提供了哪些工具、选择了哪个工具、生成了哪些参数、哪个身份执行操作、调用了哪些其他智能体、发生多少次重试，以及产生了哪些策略或评估事件。

最终答案并非充分证据。智能体可能读取未授权数据后仍返回正确句子，可能经过三次不必要的昂贵调用才完成任务，可能选择错误工具后被回退逻辑挽救，也可能因目标服务拒绝超额权限而安全失败。这些结果需要能够关联智能体、模型、检索、工具、会话、策略与成本事件的追踪。

信号	回答的问题	隐私或质量注意事项
智能体 / 运行跨度	哪个版本、端点、调用方与会话处理了任务?	不要把可猜测的用户标识符作为会话安全机制
模型调用跨度	发生了哪个模型、延迟、令牌用量与响应状态?	提示词 / 输出内容可能敏感，并且可以不捕获
检索跨度	哪个查询、来源、筛选器与文档支持了响应?	文档文本与 ACL 元数据需要保护
工具调用跨度	使用了哪个工具、参数、结果、身份与时长?	参数 / 结果可能包含秘密或个人数据
智能体间跨度	哪个智能体把什么任务委派给哪个对等智能体?	委派可能模糊权限与责任
策略 / 评估事件	哪个护栏、批准或评估器通过或失败?	分数需要解释与评估器版本
用量 / 成本指标	完成或失败的任务消耗了什么?	聚合成本必须保持可归因到版本与租户 / 工作负载
人工标注	响应在上下文中是否有用、安全或正确?	反馈可能有偏差，需要审查者来源

表 19-11 智能体可观测性遵循轨迹

OpenTelemetry 正逐渐成为这些信号的共同语义层。一致的跨度名称与属性让团队能够比较框架与提供商、关联评估结果，并把追踪导出到既有可观测性后端。内容捕获应由用户选择并接受治理。秘密、访问令牌、原始凭据和不必要的个人数据绝不能放入提示词、工具参数或追踪属性。

智能体 SLO 应组合运营与行为指标。可用性和延迟仍然必要，但并不充分。有用指标包括任务完成率、工具调用成功率、未授权操作尝试、批准放弃率、引用覆盖率、评估通过率、每项完成任务成本，以及从瞬时故障中恢复的能力。平台仪表板只有在能把这些指标归因到版本和发布，而不是展示混杂总数时才真正有价值。

19.12 评估是发布门禁

智能体评估比模型输出评估更难，因为测试对象是一条轨迹。评估者不只要考虑智能体说了什么，还要考虑它做了什么、尝试了什么、使用了哪些证据、如何处理失败，以及操作是否获准。正确的最终答案不能抵消一次被禁止的工具调用；请求操作超出权限时，安全拒绝并不是失败。

评估属于资源生命周期。测试数据集应版本化，评估器配置应具名并版本化，评估运行应识别候选智能体版本、模型部署、工具集、知识状态、运行时、裁判模型和阈值。发布决策应记录哪些门禁通过、哪些例外获准以及由谁批准。

评估维度	问题示例	可用度量
任务完成	智能体是否实现预期业务结果？	通过 / 失败量表、状态验证、人工审查
答案质量	响应是否相关、连贯且有用？	量表或经校准的裁判分数
依据性	声明是否来自合格证据？	引用准确率 / 覆盖率与无依据声明率
轨迹质量	步骤是否必要、有序且高效？	多余步骤率、循环检测、路径量表
工具选择	智能体是否选择了正确能力？	工具选择准确率与禁止工具使用率
参数正确性	参数是否有效、限定作用域且安全？	架构通过率、边界违规、试运行比较
授权行为	智能体是否尊重调用方与操作作用域？	被拒绝操作测试与权限提升尝试
安全与策略	内容和操作是否满足策略？	安全评估器、策略事件率、红队用例
稳健性	智能体如何处理缺失数据、工具故障、注入与歧义？	对抗与故障注入通过率
性能	任务是否达到延迟与可靠性目标？	端到端延迟、超时、重试、成功率
经济性	轨迹与结果是否相称？	每项完成任务成本与预算差异

表 19-12 智能体评估必须包括行为与权限



图 19-5 评估是组合测试数据、追踪、标准与人工审查的生命周期回路。

强健的评估计划使用多种数据源。设计用例覆盖要求与已知风险；反向测试验证拒绝、缺失权限、无效输入与不安全工具；合成用例在发布前扩大覆盖；生产追踪揭示真实行为，并可整理为版本化回归集；人工审查者校准模糊判断与高影响结果。

LLM 充当裁判的评估方式很有用，却不能自证有效。裁判模型可能不一致、对格式敏感，并偏爱冗长或风格相近的答案。应使用人工标注示例校准裁判输出，尽可能采用确定性检查，记录裁判模型与提示词，并避免根据一个综合分数晋级发布。发布门禁应同时包含硬阻断项（例如未授权操作或严重安全故障）与质量阈值。

门禁	阈值示例	决策语义
关键授权	强制测试集中成功的未授权操作为 0	硬阻断；未经安全审查不得例外
工具架构与作用域	100% 符合有效架构；作用域外目标为 0	硬阻断
任务完成	在已批准核心场景中至少达到 90%	质量门禁；按细分调查回归
依据性	需要证据的声明至少达到 95% 引用覆盖率	质量与审计门禁
安全	无严重内容 / 操作安全故障；较低严重度比例有界	按严重度确定硬门禁 / 条件门禁
延迟	目标环境的 P95 低于服务目标	运营门禁

门禁	阈值示例	决策语义
成本	每项完成任务成本的中位数与 P95 位于已批准包络内	FinOps 门禁
人工审查	所有高影响场景均已审查并签署	治理门禁

表 19-13 发布门禁结构示例

19.13 智能体边界的 AI 治理

AI 治理在智能体边界变得具体，因为平台把模型行为连接到组织数据与操作。NIST AI RMF 的治理、映射、度量和管理职能可以在此落地。治理分配所有者、角色、风险容忍度与审查节奏；映射识别用户、数据、工具、影响与依赖；度量使用评估、遥测、红队测试与事件证据；管理应用发布门禁、批准、限制、缓解措施与退役决策。

治理策略应与影响成比例。只读文档智能体与更改访问角色的智能体不应共享同一发布流程。应按数据敏感度、操作权限、自主性、持久性、外部通信和爆炸半径对智能体分类。分类决定所需身份模型、网络控制、人工批准、评估覆盖、事件响应与监控。

风险源	控制平面控制	证据
目标或提示词劫持	指令层级、不可信内容标记、工具允许列表、高风险批准	注入测试结果与被阻止工具追踪
工具误用或权限滥用	限定作用域身份、目标侧授权、操作类别、试运行、批准绑定	角色分配、拒绝测试、批准记录
不安全委派	显式子智能体允许列表、有界上下文与工具、A2A 身份验证	智能体图与委派追踪
记忆污染或隐私泄露	记忆作用域、来源、同意、纠正、删除、保留	记忆策略与删除测试
知识污染	来源准入、摄取验证、血缘、检索评估	来源清单与评估报告
敏感追踪披露	内容捕获策略、脱敏、独立追踪读取者角色、保留	追踪配置与访问审查
无界消耗	运行限制、工具 / 时间预算、配额、循环检测、成本警报	容量策略与每任务成本仪表板
不安全的产品成熟度假设	预览功能清单、生产例外流程、迁移 / 回退计划	功能状态登记册与批准
事件期间失去控制	终止开关、禁用端点、撤销工具、终止会话、回滚	事件演练与退役运行手册

表 19-14 智能体治理把风险转化为资源控制

产品状态：预览状态是一项治理属性。应记录哪些生产依赖处于预览、受区域限制或计划退役，并要求明确例外或回退计划。

19.14 成本、配额、容量与价值单位

智能体成本是路径成本。一次运行可以消耗模型输入与输出令牌、检索与重排序、工具 API、托管运行时、代码沙箱、记忆操作、网络出站、可观测性摄取、评估裁判与人工审查。每令牌价格无法解释反复调用昂贵工具或让托管会话闲置的智能体经济性。

有用的单位是每项已完成且获接受任务的成本。这一分母会暴露失败、重试与低质量轨迹。两个版本可能具有相同模型令牌成本，但其中一个完成两倍任务、调用更少工具或产生更少批准拒绝。因此，发布门禁应在质量和安全之外加入成本与容量证据。

成本组成	计量单位或驱动因素	控制平面响应
模型推理	输入 / 输出 / 缓存令牌或预配容量	模型 / 部署选择、提示词 / 上下文策略、配额
知识	摄取、存储、检索、重排序、嵌入	刷新策略、top-k、索引层、来源生命周期
工具与连接器	API 调用、事务、外部服务单位	允许列表、预算、批处理、缓存、速率限制
托管运行时	CPU / GPU、内存、会话时长、沙箱时间	自动缩放、空闲超时、运行时类别、会话过期
记忆	写入、提取、存储、检索	作用域、保留、总结、删除
可观测性	追踪 / 日志 / 指标摄取与保留	抽样、内容策略、保留、导出
评估	裁判调用、测试量、合成数据、人工审查	分层套件、计划节奏、基于风险的覆盖
失败与返工	重试、循环、放弃、回滚、支持	循环防护、回退、事件学习、回归测试

表 19-15 智能体经济性比令牌经济性更广

```
cost_per_accepted_task = (
    inference + retrieval + tools + runtime + memory
    + observability + evaluation + human_review
```

```
) / accepted_completed_tasks
```

示例 19-4：智能体运维的单位经济性分母

19.15 Azure 参考控制平面：Microsoft Foundry Agent Service

Microsoft Foundry 是有用的参照，因为它让智能体生命周期跨项目、智能体、版本、端点、身份、工具、会话、追踪、评估与发布变得可见。截至 2026 年 8 月，其 Agent Service 区分基于提示词的智能体与 Hosted agents。基于提示词的智能体是组合 Foundry 模型、指令、工具和提示词的声明式定义；Hosted agents 把自定义代码或受支持框架打包到平台托管环境。两者都可以调用、观测、评估、版本化与发布，但赋予作者的责任不同。

Foundry 项目是连接模型、智能体定义、工具箱、知识、评估数据集与 Application Insights 的治理边界。智能体版本提供不可变发布状态；稳定端点把调用方与版本变更解耦。托管会话为逻辑工作负载提供有状态隔离沙箱，其生命周期和空闲超时行为由平台管理。对话与响应表示数据平面交互状态。

身份横跨多个主体。用户与部署流水线需要用于控制平面和数据平面操作的 Foundry 角色；最小权限使用者角色可以调用智能体，但不能创建智能体；项目拥有访问依赖项的托管身份；智能体还可以获得一个用于限定作用域工具访问的专用 Microsoft Entra 身份。工具箱整合工具与身份验证模式，包括智能体身份、项目身份、基于密钥的访问和 OAuth 代表用户传递。私有网络配置可在当前功能支持范围内隔离模型、数据与托管智能体流量。

Foundry 对象或能力	治理角色
Foundry 账户与项目	组织资源、角色、网络、模型部署、智能体、工具与证据
基于提示词的智能体	声明式指令、模型、工具、知识与记忆绑定
Hosted agent	为基于代码或框架的智能体实现提供托管环境
智能体版本	具有状态与生命周期操作的不可变可部署发布
已发布端点 / 应用	稳定调用与共享界面
工具箱与 MCP / OpenAPI 工具	智能体能力的托管发现与身份验证边界
智能体身份	限定作用域资源与工具访问所用专用主体
会话、对话、响应	运行时交互的状态与执行对象

Foundry 对象或能力	治理角色
Application Insights 追踪	面向模型、工具与智能体、基于 OpenTelemetry 的轨迹证据
评估数据集、标准、运行	候选或生产行为的版本化质量与安全证据
智能体监控仪表板	生产流量的运营与评估指标

表 19-16 作为参考对象模型的 Microsoft Foundry Agent Service

追踪使用 OpenTelemetry 约定，记录用户输入和智能体输出、模型与工具操作、令牌用量、延迟及多智能体交互。评估可以针对版本化数据集运行候选智能体，应用智能体特定量表与内置安全或质量评估器，再把结果连接到项目。生产追踪可以整理成评估数据集，从而闭合学习回路。

产品状态必须保持明确。Foundry 的功能并非都具有相同的正式发布状态、区域支持或私有网络支持。工作流能力与预览功能可能快速变化。生产设计应保留功能状态登记册，验证当前区域和网络限制，并避免让未经批准的预览功能成为隐藏关键依赖。

19.16 多云转换：保留控制平面意图

本书中的四个平台不会公开一种通用智能体对象模型。正因如此，比较应以工程意图而非截图相似度为基础。稳定问题是：定义是否受治理，工具和知识是否明确绑定，是否存在身份和工作区边界，运行时与会话是否受管理，追踪与评估是否支持发布决策，以及成本和删除是否保持可见。



图 19-6 智能体平台比较应使用共同控制平面视角，而不是强求功能对等。

19.16.1 阿里云：Model Studio 与 PAI-LangStudio

阿里云目前公开了不止一个相关产品界面。Model Studio 文档描述了组合模型、提示词、知识库、插件或 MCP 服务与记忆的智能体应用和工作流应用。智能体应用允许模型驱动决策；工作流应用编码可视化节点与转换。应用可以测试、发布，并通过 DashScope SDK 或 HTTP 调用。

官方文档还把 Model Studio 的“应用开发”标签页及相关应用、组件与数据功能标记为预览，在某些国际区域场景中受访问限制。这一状态本身就是对象模型的一部分：生产架构必须记录应用在哪里可用、接口是否对新租户受支持，以及存在哪条迁移或回退路径。

PAI-LangStudio 是更适合企业生命周期比较的产品。它提供低代码与高代码智能体开发、知识、MCP 与工具、追踪、评估、通过 PAI 服务进行部署、开发与生产隔离、RAM 与工作区 RBAC、专用 VPC 与计算选项，以及全链路可观测性。不能把两个界面压缩成一个虚构产品。Model Studio 适合以模型为中心的智能体与工作流应用；LangStudio 又增加了适合企业智能体工程的工作区与部署边界。

19.16.2 腾讯云 ADP

腾讯云智能体开发平台在企业与工作区边界内组织智能体应用。其已记录的应用模式包括标准智能体行为、单工作流、多智能体协调和其他引导模式。工作区提供平台侧功能与数据权限，应用侧用户权限则可以限制终端用户可用的知识作用域。这种分离很重要：构建者授权与终端用户数据资格是不同控制。

ADP 公开知识库、连接器与工具、工作流、多智能体结构、长期记忆、应用发布、应用分析、评估、操作日志与警报管理。工作流可以引用已发布或尚未发布的子工作流，异常行为包括重试与备用流。应用分析和警报公开调用成功率、令牌用量、QPM 或 TPM 等运营指标。评估支持智能体应用的批量测试。

工程经验在于，工作区、应用模式、知识作用域、发布状态、终端用户权限、操作日志、评估结果与警报策略是托管契约中彼此不同的部分。比较不能把 ADP 简化成“聊天机器人构建器”。

19.16.3 火山引擎 AgentKit

在这里考察的中国云平台中，火山引擎 AgentKit 展示的资源目录分解得最明显。其文档把托管运行时、集成工具沙箱、长期记忆、多模态知识、MCP 网关、技能、身份与凭据管理、权限、端到端可观测性、评估和生命周期日志列为平台能力。

运行时独立于框架与模型，可以随流量缩放。工具可在托管沙箱中执行。记忆与知识是不同托管资源。MCP 网关提供共同连接界面。可观测性包含针对 AgentKit 运行时调用、基于 OpenTelemetry 的追踪与指标。评估由数据集、实验、会话分析与标注表示。计费文

档把运行时、沙箱 / 工具和网关组件分开，使运营成本边界明确。

这种分解对教学很有价值，因为它表明企业智能体由托管依赖项组装而成。治理任务是决定哪些资源共享工作区与身份边界、如何版本化和观测，以及哪些内容仍在 AgentKit 外部。

稳定视角	Microsoft Foundry	阿里云	腾讯云 ADP	火山引擎 AgentKit
项目边界	Foundry 账户 / 项目	PAI 工作区；Model Studio 应用界面	企业加工作区	AgentKit 账户 / 项目资源
智能体定义	基于提示词或 Hosted agent	智能体 / 工作流应用；LangStudio 应用流	标准 / 工作流 / 多智能体模式的应用	部署到运行时的智能体代码 / Studio 定义
运行时	托管提示词运行时或 Hosted agent 会话	Model Studio 托管应用；EAS / LangStudio 运行时	ADP 托管应用运行时	托管 Agent Runtime
工具	工具目录、工具箱、MCP、OpenAPI、内置工具	插件、MCP、工具、LangStudio 组件	连接器与工具、技能广场	工具沙箱、MCP 网关、技能
知识	Foundry 知识 / 搜索集成	Model Studio 或 LangStudio 知识库	带终端用户作用域控制的 ADP 知识库	Viking / AgentKit 知识资源
记忆 / 状态	对话、会话、托管记忆	应用记忆 / LangStudio 上下文组件	长期记忆与会话 ID	托管记忆存储与会话
身份	Entra、Foundry RBAC、项目与智能体身份、OBO	RAM、PAI 工作区 RBAC、服务角色	CAM 加平台侧与应用侧权限	智能体身份、凭据、权限网关
证据	Application Insights 追踪与监控	SLS / 托管 OTel 追踪、日志、评估	分析、操作日志、评估、警报	OTel 追踪 / 指标、日志、评估实验
生命周期	版本、发布、端点、会话、删除	测试 / 发布 / 调用；LangStudio 部署与环境分离	草稿 / 已发布状态、工作流复用、应用发布	运行时创建 / 更新、工具 / 记忆 / 知识生命周期
经济性	模型、运行时、工具、可观测性、评估计量项	模型加 PAI / EAS / OSS / SLS / OTel 资源	套餐 / 后付费资源与用量指标	运行时、沙箱 / 工具、网关、模型与相关资源

表 19-17 当前智能体平台对象模型映射（2026 年 8 月）

比较规则：本表记录当前文档所述产品界面，并非声称功能对等。提供商没有公开可比托管对象时，应在架构记录中标注“外部”“预览”“区域受限”或“未验证”。

能力检查 C5：智能体平台对象模型比较

C5.1 能力目标

基于证据比较 Microsoft Foundry、阿里云、腾讯云 ADP 和火山引擎 AgentKit 如何表示一个假想智能体应用。本练习不要求购买或部署资源，目的在于证明你能够识别控制平面对象、生命周期状态、权限边界、可观测性、评估、成本与缺口，而不强求虚假等价。

使用“云成本研究助手”场景。该助手读取已批准账单导出与云文档，解释异常，并提出但不执行优化操作。其工具全部只读；它拥有组织知识，不使用跨用户长期记忆，并设置针对依据性、工具选择、延迟与成本的发布门禁。

C5.2 前提条件与成本边界

- 可以访问所有四个平台当前的官方文档；门户访问可选。
- 用于存储证据包的存储库或文件夹。不得在证据包中放入访问令牌、账户 ID 或专有截图。
- 不需要计费部署。如果选择检查现有租户，不要仅为本检查创建运行时、模型、知识或评估资源。
- 为每个提供商记录核验日期与区域，因为产品状态与对象名称可能变化。

C5.3 创建证据包结构

```
c5-agent-platform-comparison/
  scenario.md
  common-contract.yaml
  microsoft-foundry.yaml
  alibaba-cloud.yaml
  tencent-adp.yaml
  volcengine-agentkit.yaml
  lifecycle-comparison.md
```

```
  identity-and-tool-boundaries.md
```

```
  evaluation-gate.md
```

```
  gaps-and-preview-register.md
```

```
  acceptance-checklist.md
```

示例 19-5：证据包结构

C5.4 定义共同契约

```
scenario: Cloud Cost Research Assistant
```

```
impact: read-only recommendation
```

```
required_objects:
```

```
  - project_or_workspace
```

```
  - agent_definition
```

```
  - immutable_version_or_release_state
```

```
  - model_binding
```

```
  - knowledge_binding
```

```
  - read_only_toolset
```

```
  - caller_and_agent_identity
```

```
  - endpoint_or_application_call_contract
```

```
  - session_or_conversation_boundary

  - trace_and_operation_log

  - evaluation_dataset_and_run

  - quota_cost_and_retirement_evidence

release_gates:

  unauthorized_successful_actions: 0

  grounded_core_answers: ">= 95%"

  correct_tool_selection: ">= 95%"

  p95_latency: documented_target

  cost_per_accepted_task: documented_envelope
```

示例 19-6：四平台比较的共同契约

C5.5 如实分类每个平台对象

对每项所需对象，记录提供商产品名称和以下五类之一：托管资源、嵌入式配置、外部依赖项、预览或区域受限能力、未验证。包括官方来源 URL 和访问日期。“未验证”是有效结果，而且通常比虚构等价项更有用。

分类	含义	示例
托管资源	具有提供商管理生命周期的具名或可寻址对象	智能体运行时、工作区、知识库、评估运行
嵌入式配置	较大托管对象内的字段	应用内的系统指令或工具绑定
外部依赖项	由其他服务治理的能力	Application Insights、OSS、EAS、外部 MCP 服务器

分类	含义	示例
预览 / 区域受限	已有文档，但尚未普遍达到生产就绪或可用状态	预览应用开发界面或网络限制
未验证	没有找到当前一手来源证据	不要根据营销材料或相邻产品推断

表 19-18 能力检查 C5 的分类词汇

C5.6 绘制每个平台的生命周期

识别候选对象如何从草稿或源代码进入测试、评估、发布、调用、监控、修订、回滚和退役。记录哪些步骤有明确平台对象，哪些依赖外部 CI/CD 或运营流程。注明删除后果：版本被移除、会话终止、端点禁用、外部知识保留或身份残留。

C5.7 追踪身份与工具路径

对每个平台，识别构建者身份、发布身份、应用调用方、终端用户上下文、智能体或项目身份、工具身份验证模式与目标服务授权。确认只读工具可以独立于智能体构建者限定作用域。记录支持或要求的用户委派、应用身份、智能体身份、密钥或无身份验证工具。

C5.8 指定评估门禁

定义至少十个测试用例：五个正常成本分析问题、两个缺失数据用例、一个嵌入账单备注中的提示词注入用例、一个请求修改操作的尝试，以及一个工具故障用例。把每个用例映射到平台评估能力；若没有记录托管等价项，则映射到外部评估器。记录数据集版本、候选版本、标准、阈值与发布决策。

C5.9 生成缺口与预览登记册

字段	必填项
能力或依赖项	确切产品对象或功能
状态	正式发布、预览、区域受限、已弃用、外部或未验证
运营影响	能力变化时，哪些内容会失败或转为人工操作？
回退	替代提供商服务、外部实现或缩小作用域的设计
复审日期	必须何时重新检查状态
所有者	对决策负责的角色

表 19-19 缺口与预览登记册

C5.10 验收标准

标准	通过条件
对象模型	所有四个平台的全部必需契约行均已分类
一手证据	每项产品声明都有官方来源 URL 与核验日期
不等价性	明确说明至少一项真实差异或未验证缺口
生命周期	已映射草稿、测试、评估、发布、调用、观测、修订与退役
身份	构建者、调用方、智能体 / 项目、工具与目标授权相互分离
评估	版本化的十用例门禁包括轨迹与授权检查
成本	已识别模型、运行时、工具、知识、可观测性与评估成本边界
安全	智能体保持只读；修改请求被拒绝或仅路由到建议
保留	区分会话、记忆、知识、追踪与评估保留
卫生	证据包不包含秘密、专有租户标识符或原始敏感对话

表 19-20 能力检查 C5 验收标准

C5.11 清理与刷新

默认练习不创建云资源，因此清理内容是删除含租户特定详情的本地临时文件，只保留已净化证据包。设置不晚于完成后六个月的刷新日期；若提供商宣布重大智能体平台迁移，则应提前。将比较用于架构或采购决策前，请重新检查产品状态。

19.17 CloudDesk 复用：受治理智能体平台适配器

CloudDesk FinOps Agent 以后将实现自己的无服务器运行时、租户控制平面、批准和跨租户隔离。本章不构建该系统。可复用输出是一份适配器契约，使 CloudDesk 可以检查或使用托管智能体平台能力，而不把任何提供商视为不透明聊天机器人端点。

能力	用途	风险类别	所需证据
agent.definition.inspect	返回智能体类型、模型、工具、知识、记忆、身份、版本、端点和状态，但不返回秘密	R0	提供商 ID、实际版本、依赖图、来源时间戳
agent.release.compare	比较两个版本并识别行为、权限、成本与产品状态变化	R1	发布清单、差异、评估摘要、回滚目标

能力	用途	风险类别	所需证据
agent.trace.read	为已授权工作负载检索经过隐私筛选的轨迹	R0 / R1	追踪 ID、版本、工具跨度、脱敏状态、调用方作用域
agent.evaluation.run	运行已批准数据集并返回阈值结果	R1	数据集 / 评估器版本、分数、解释、成本、决策
agent.tool.permission.audit	比较已公开工具与目标侧角色和批准规则	R1	工具目录、身份验证模式、角色分配、被拒绝操作测试

表 19-21 面向 CloudDesk 的托管智能体平台能力

这些能力应返回结构化证据，而不只是自然语言。agent.definition.inspect 不得泄露 API 密钥、连接秘密或原始记忆；agent.trace.read 必须在查询提供商前执行工作负载与租户作用域；agent.evaluation.run 可能产生成本，应要求已批准数据集与预算；agent.tool.permission.audit 可以建议收窄作用域，但不得创建角色分配——权限变更属于需要明确批准的确定性管理工作流。

19.18 要点

- 提示词加工具循环是一种应用机制。受治理智能体资源加入身份、生命周期、网络、策略、证据、成本与退役语义。
- 智能体定义、版本、端点、运行时、会话、对话、运行、工具、知识、记忆、追踪与评估是不同对象，应保持可区分。
- 基于提示词、托管、工作流和多智能体形态是不同运营契约。使用能够满足要求的最低自主性形态。
- 构建或发布智能体的控制平面权限，应与调用智能体或执行工具的数据平面权限分开。
- 工具描述指导模型选择；目标侧身份验证与授权才授予权限。高风险操作所需的是更强控制，而不是更好的提示词。
- 知识、记忆、会话状态、对话历史、事务状态与追踪内容需要不同的作用域、保留、纠正和删除策略。
- 智能体发布包括实际模型、工具、知识、记忆策略、运行时、安全控制和评估证据，而不只是指令。
- 可观测性必须跨模型、检索、工具、智能体、策略、成本与人工反馈追踪轨迹，同时保护敏感内容。
- 评估是发布门禁。它必须包括任务完成、轨迹、工具、授权、安全、性能与成本，并把严重违规作为硬阻断项。
- 如实记录托管资源、嵌入式配置、外部依赖、预览状态和未验证缺口时，多云比较最为有力。

- 当平台把智能体行为与依赖项转化为可问责、可运营、可逆转的云资源生命周期时，它就成为 AI PaaS 的一部分。

参考文献

[1] 美国国家标准与技术研究院：《人工智能风险管理框架（AI RMF 1.0）》。https://doi.org/10.6028/NIST.AI.100-1 访问于 2026 年 8 月 25 日。

[2] 美国国家标准与技术研究院：《人工智能风险管理框架：生成式人工智能概况（NIST AI 600-1）》。https://doi.org/10.6028/NIST.AI.600-1 访问于 2026 年 8 月 25 日。

[3] NIST AI 资源中心：AI RMF 核心——治理、映射、度量与管理。https://airc.nist.gov/airmf-resources/airmf/5-sec-core/ 访问于 2026 年 8 月 25 日。

[4] Microsoft Learn：《什么是 Microsoft Foundry Agent Service？》。https://learn.microsoft.com/en-us/azure/foundry/agents/overview 访问于 2026 年 8 月 25 日。

[5] Microsoft Learn：Microsoft Foundry 中的智能体开发生命周期。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/development-lifecycle 访问于 2026 年 8 月 25 日。

[6] Microsoft Learn：Microsoft Foundry Agent Service 中的 Hosted agents。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/hosted-agents 访问于 2026 年 8 月 25 日。

[7] Microsoft Learn：Microsoft Foundry 中的智能体身份概念。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/agent-identity 访问于 2026 年 8 月 25 日。

[8] Microsoft Learn：Hosted agent 权限参考。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/hosted-agent-permissions 访问于 2026 年 8 月 25 日。

[9] Microsoft Learn：Foundry Agent Service 中的工具类型。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/tool-catalog 访问于 2026 年 8 月 25 日。

[10] Microsoft Learn：在 Foundry 中创建和管理工具箱。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/tools/toolbox 访问于 2026 年 8 月 25 日。

[11] Microsoft Learn：Microsoft Foundry 中的工具箱身份验证。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/tools/tool-authentication 访问于 2026 年 8 月 25 日。

[12] Microsoft Learn：把智能体连接到 MCP 服务器端点。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/tools/model-context-protocol 访问于 2026 年 8 月 25 日。

[13] Microsoft Learn：智能体追踪概览。https://learn.microsoft.com/en-us/azure/foundry/observability/concepts/trace-agent-concept 访问于 2026 年 8 月 25 日。

[14] Microsoft Learn：评估 AI 智能体。https://learn.microsoft.com/en-us/azure/foundry/observability/how-to/evaluate-agent 访问于 2026 年 8 月 25 日。

[15] Microsoft Learn：把智能体追踪转换为评估数据集。https://learn.microsoft.com/en-us/azure/foundry/observability/how-to/traces-to-dataset 访问于 2026 年 8 月 25 日。

[16] Microsoft Learn：使用 Agent Monitoring Dashboard 监控智能体。https://learn.microsoft.com/en-us/azure/foundry/observability/how-to/how-to-monitor-agents-dashboard 访问于 2026 年 8 月 25 日。

[17] Microsoft Learn：管理 Hosted agent 会话。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/manage-hosted-sessions 访问于 2026 年 8 月 25 日。

[18] Microsoft Learn：按用户隔离 Hosted agent 会话。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/isolate-sessions-per-user 访问于 2026 年 8 月 25 日。

[19] Microsoft Learn：部署和管理 Hosted agent。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/deploy-hosted-agent 访问于 2026 年 8 月 25 日。

[20] Microsoft Learn：Microsoft Foundry 门户正式发布概览。https://learn.microsoft.com/en-us/azure/foundry/concepts/general-availability 访问于 2026 年 8 月 25 日。

[21] 阿里云文档：Model Studio 中的智能体与工作流应用类型。https://www.alibabacloud.com/help/en/model-studio/application-introduction 访问于 2026 年 8 月 25 日。

[22] 阿里云文档：Model Studio 中的智能体应用。https://www.alibabacloud.com/help/en/model-studio/single-agent-application 访问于 2026 年 8 月 25 日。

[23] 阿里云文档：Model Studio 中的工作流应用。https://www.alibabacloud.com/help/en/model-studio/workflow-application/ 访问于 2026 年 8 月 25 日。

[24] 阿里云文档：Model Studio 插件概览。https://www.alibabacloud.com/help/en/model-studio/plug-in-overview 访问于 2026 年 8 月 25 日。

[25] 阿里云文档：使用 API 调用 Model Studio 应用。https://www.alibabacloud.com/help/en/model-studio/application-calling-guide 访问于 2026 年 8 月 25 日。

[26] 阿里云 PAI 文档：使用 LangStudio 构建企业 AI 智能体。https://www.alibabacloud.com/help/en/pai/langstudio-1/ 访问于 2026 年 8 月 25 日。

[27] 阿里云 PAI 文档：为 LangStudio 授予权限。https://help.aliyun.com/en/pai/user-guide/cloud-product-dependency-and-authorization-langstudio 访问于 2026 年 8 月 25 日。

[28] 阿里云 PAI 文档：创建和管理工作区。https://www.alibabacloud.com/help/en/pai/create-and-manage-workspaces 访问于 2026 年 8 月 25 日。

[29] 阿里云 PAI 文档：角色与权限。https://www.alibabacloud.com/help/en/pai/appendix-list-of-roles-and-permissions 访问于 2026 年 8 月 25 日。

[30] 腾讯云文档：智能体开发平台产品概览。https://www.tencentcloud.com/document/product/1254/69956 访问于 2026 年 8 月 25 日。

[31] 腾讯云文档：智能体应用与四种模式。https://www.tencentcloud.com/document/product/1254/73172 访问于 2026 年 8 月 25 日。

[32] 腾讯云文档：《什么是多智能体模式？》。https://www.tencentcloud.com/document/product/1254/73337 访问于 2026 年 8 月 25 日。

[33] 腾讯云文档：企业、工作区与权限概览。https://www.tencentcloud.com/document/product/1254/73349 访问于 2026 年 8 月 25 日。

[34] 腾讯云文档：平台侧用户权限。https://www.tencentcloud.com/document/product/1254/73347 访问于 2026 年 8 月 25 日。

[35] 腾讯云文档：应用侧用户权限。https://www.tencentcloud.com/document/product/1254/72504 访问于 2026 年 8 月 25 日。

[36] 腾讯云文档：长期记忆。https://www.tencentcloud.com/document/product/1254/73194 访问于 2026 年 8 月 25 日。

[37] 腾讯云文档：应用分析。https://www.tencentcloud.com/document/product/1254/72572 访问于 2026 年 8 月 25 日。

[38] 腾讯云文档：应用评估。https://www.tencentcloud.com/document/product/1254/72510 访问于 2026 年 8 月 25 日。

[39] 腾讯云文档：操作日志。https://www.tencentcloud.com/document/product/1254/77451 访问于 2026 年 8 月 25 日。

[40] 腾讯云文档：警报管理。https://www.tencentcloud.com/document/product/1254/78648 访问于 2026 年 8 月 25 日。

[41] 火山引擎文档：AgentKit 文档中心。https://docs.volcengine.com/docs/86681?lang=zh 访问于 2026 年 8 月 25 日。

[42] 火山引擎文档：《什么是 AgentKit？》。https://docs.volcengine.com/docs/86681/1844823?lang=zh 访问于 2026 年 8 月 25 日。

[43] 火山引擎文档：AgentKit 运行时。https://docs.volcengine.com/docs/86681/2605800?lang=zh 访问于 2026 年 8 月 25 日。

[44] 火山引擎文档：AgentKit 插桩字段与指标。https://docs.volcengine.com/docs/86681/2222166?lang=zh 访问于 2026 年 8 月 25 日。

[45] 火山引擎文档：AgentKit 评估工具概览。https://docs.volcengine.com/docs/86681/2607693?lang=zh 访问于 2026 年 8 月 25 日。

[46] 火山引擎文档：AgentKit 计费项。https://docs.volcengine.com/docs/86681/2480915?lang=zh 访问于 2026 年 8 月 25 日。

[47] Model Context Protocol：授权规范与安全注意事项。https://modelcontextprotocol.io/specification/draft/basic/authorization 访问于 2026 年 8 月 25 日。

[48] Model Context Protocol：安全最佳实践。https://modelcontextprotocol.io/docs/draft/tutorials/security/security_best_practices 访问于 2026 年 8 月 25 日。

[49] OpenTelemetry：语义约定。https://opentelemetry.io/docs/specs/semconv/ 访问于 2026 年 8 月 25 日。

[50] OpenTelemetry：借助语义约定实现生成式 AI 可观测性。https://opentelemetry.io/blog/2026/genai-observability/ 访问于 2026 年 8 月 25 日。

[51] OWASP GenAI 安全项目：智能体式 AI——威胁与缓解措施。https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/ 访问于 2026 年 8 月 25 日。

[52] OWASP GenAI 安全项目：2026 年智能体应用十大风险。https://genai.owasp.org/resource/owasp-top-10-for-agentic-applications-for-2026/ 访问于 2026 年 8 月 25 日。

从云能力到受治理智能体工具

前五篇建立了可以观察、验证和治理的云资源。最后一篇把这些能力交给智能体使用，逐项限制工具能读取什么、能计划什么、何时可以执行，以及如何证明实际效果。

20 将云能力转化为智能体工具

第六篇 / 第 20 章

将云能力转化为智能体工具

从提供商 API 走向安全、版本化、可审计的能力契约

智能体工具不只是带描述的 API 操作，而是一份受治理契约，把模型可见语义与身份、作用域、验证、风险、批准、证据、成本、生命周期及提供商特定执行绑定起来。

本章内容

- 为何云提供商 API、SDK、CLI 命令或连接器还不属于对智能体安全的工具
- 如何定义保留资源语义和提供商真实状态的、与提供商无关的能力契约
- 为何工具描述与架构引导模型选择，却不会授予权限或证明效果
- 如何选择工具粒度，并避免不受限制的 CLI、Shell、SQL 或 REST 执行等危险通用工具
- 如何把观测、建议、规划、执行、验证与回滚分成不同权限阶段
- 为何调用方身份、工作负载身份、委派访问、租户作用域与目标服务授权必须保持明确
- JSON Schema、OpenAPI、函数调用、MCP 和工作流连接器如何融入受治理工具架构
- 如何在模型生成后验证语法、语义、前置条件、并发、成本与组织策略
- 风险类别、试运行、不可变计划摘要与人工批准如何约束修改操作
- 幂等性、异步操作、结果信封、追踪、审计记录与成本归因如何形成运营证据
- Azure、阿里云、腾讯云与火山引擎如何通过不同平台对象公开相近的工具集成意图
- 如何通过编写和审查五张供 CloudDesk 等后续系统使用的云能力卡，完成能力检查 C6

本书前几章把云服务视为可编程控制平面。虚拟机、预算、私有端点、模型部署、知识库或智能体定义，并不只是应用使用的东西，而是具有身份、生命周期、策略、遥测、成本与删除语义的资源。最后一步是在不丢弃这些语义的情况下，向智能体公开选定能力。

这一步常被实现得过于随意。团队拿来一个云 SDK 方法，写一段友好描述，通过函数调用或 MCP 服务器公开，然后断定它已经成为智能体工具。结果在演示中也许有效，却可能让概率式规划器访问宽泛凭据，接受未经验证的模型生成标识符，重试破坏性操作，返回非结构化成功字符串，并且不留下证明所请求效果确实发生的证据。

受治理工具架构不会让模型成为云控制平面。模型提出意图，并从有界能力中选择；策略与执行层根据当前状态解析意图，对相关主体进行身份验证，验证业务不变量，在需要时取得批准，调用限定作用域的提供商操作，验证效果并记录证据。目标云服务仍是最终授权机构。

本章是本项目两本书之间的交接。它通过把已验证云能力转化为安全工具契约，完成《云服务与 AI PaaS》的控制平面论证。本章不实现自定义智能体运行时、租户控制平面、桌面批准协议或跨租户隔离系统；这些机制属于《无服务器 AI 智能体开发》。这里产出的是此类运行时可以依赖的受治理能力目录。

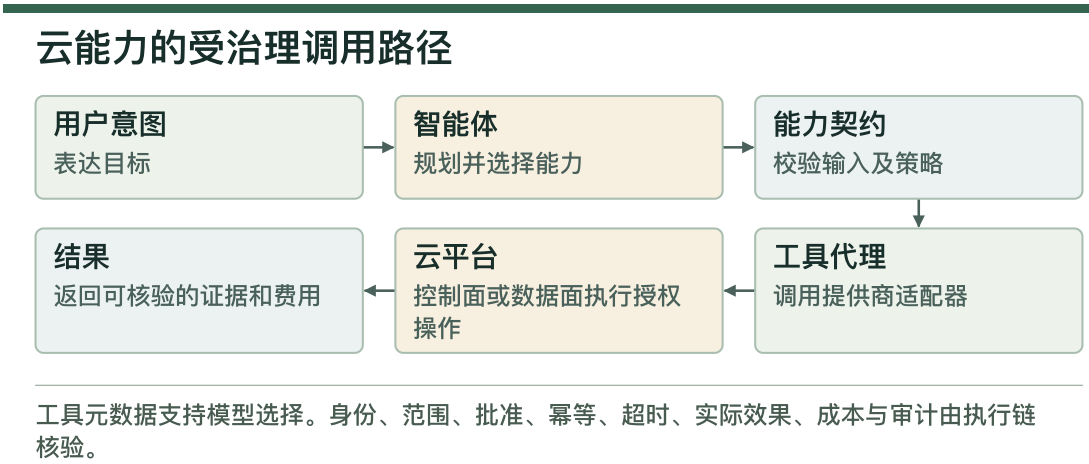


图 20-1 受治理工具边界保留模型意图、策略、执行、提供商权限与证据之间的区别。

20.1 提供商 API 不是智能体工具

云 API 面向理解服务契约的作者所编写的确定性软件，往往公开宽泛对象模型、可选参数、预览版本、长时间运行操作、提供商特定错误码，以及权限远超单项业务任务的凭据。这些特性适合基础设施自动化，但直接交给语言模型非常危险。

智能体工具的用途更窄。它应表达一项可识别的业务或工程能力，只接受该能力所需信息，显现作用域与副作用，并返回可独立验证的结果。即使模型选择意外参数组合、重复请求、误解资源名称，或通过检索内容接收恶意指令，工具仍必须安全可用。

区别不在传输方式。同一操作可以公开为本地函数、OpenAPI 端点、MCP 工具、无服务器函数、工作流连接器或内置平台工具。传输方式决定如何发现和调用操作；治理决定操作意味着什么、谁可以使用，以及如何约束其效果。

表 20-1 提供商操作与智能体工具解决不同问题

维度	原始提供商操作	受治理智能体工具
用途	公开服务 API 或 SDK 界面	表达一项有界工程意图
调用方假设	确定性代码与了解情况的开发者	概率式规划器与可能不可信的上下文
参数	按提供商塑造，并且通常很广泛	最少、类型化、按领域塑造
权限	所提供凭据允许的一切	明确主体、目标作用域与策略决策
副作用	由 API 文档定义	经过分类、限制、批准与验证
重试	通常留给客户端	幂等性与并发行为是契约的一部分
结果	提供商响应或异常	稳定结果信封加证据与警告
生命周期	API 版本与提供商生命周期	能力版本、审查、发布、弃用、退役

设计规则：绝不能把不受限制的云 CLI、Shell、SQL 控制台、通用 HTTP 客户端或原始 SDK 客户端作为生产智能体工具公开。应封装具体意图，并在目标服务保留提供商授权。

20.2 受治理能力契约

核心制品是云能力卡。它既是本书的编辑工具，也是后续智能体系统的工程契约。能力卡记录应在提供商变化后继续保持稳定的意图，并指向如实实现该意图的提供商特定适配器。

可用的概括是：智能体工具 = 能力语义 + 类型化架构 + 权限绑定 + 执行策略 + 证据契约 + 生命周期。省略任何一项都会形成隐式依赖：省略权限，运行时会退回环境凭据；省略证据，成功消息会取代证明；省略生命周期，API、策略或业务流程变化后，智能体仍会继续调用工具。



图 20-2 工具协议公开操作；能力契约提供外围治理层。

```
capability:
  id: finops.budget.create
  version: 1.0.0
  purpose: Create a scoped budget and notification policy.
  owner: cloud-finops-platform
  mode: mutating
  riskClass: R2
contract:
  inputSchema: schemas/finops.budget.create.input.json
```

```
  outputSchema: schemas/tool-result-envelope.json

  targetScope: subscription | resource-group | project

  preconditions:

    - caller may manage budgets at the resolved scope

    - amount is positive and currency matches billing scope

    - notification recipients are approved destinations

  authority:

    executionPrincipal: workload-identity:finops-budget-writer

    requiredActions: [budget.read, budget.write]

  execution:

    dryRunRequired: true

    approvalRequired: true

    idempotencyKey: required

    timeout: PT2M

  evidence:

    emit: [trace, policyDecision, providerOperationId, verificationRead,
costScope]
```

```
  lifecycle:
    status: published
    deprecationDate: null
```

示例 20-1：与提供商无关的云能力卡

表 20-2 云能力卡的最低字段

字段组	必答问题	重要原因
发现	稳定 ID、用途、所有者、版本与状态是什么？	支持整理、复用、审查与退役。
语义	支持哪项确切意图？明确排除哪些内容？	防止通用或模糊权限。
架构	接受什么输入与输出结构？	约束模型生成的参数并稳定调用方。
权限	哪个主体行动？在哪个目标作用域？通过哪条委派或工作负载路径？	防止环境权限与权限过大的凭据。
执行	工具是只读、可逆、修改、异步还是破坏性？	决定策略、重试、批准与回滚。
证据	发出哪些追踪、提供商操作、验证读取、策略结果与成本字段？	让效果可审查、可归因。
生命周期	如何发布、更改、弃用与退役能力？	阻止静默行为漂移。

20.3 按意图而不是 API 界面选择工具粒度

工具粒度是一项安全决策。工具过宽会迫使模型合成低层提供商命令，并提高触及无关功能的概率；工具过窄则会产生数百项几乎无法区分的操作，使选择变得不可靠。合适边界通常是一项具有主要效果的稳定工程意图。

良好的能力名称是在受治理名词上施加动词，例如 cloud.resource.inventory、network.exposure.audit、finops.budget.create、gitops.change.propose、rag.knowledge.refresh。它们说明操作用途，而不暴露 az、aliyun、tccli、ve、REST 路径、SDK 类或 Shell 语法等实现细节。

读取路径和写入路径应当分开。资源清单工具不应包含删除标志；建议工具不应静默应用建议；GitOps 提案工具可以创建供审查的拉取请求，而不是直接修补集群。这种分离让权限可以组合：智能体可以广泛观测、在某个领域内提出建议，而只通过少量经过审查的操作执行修改。

表 20-3 保留或破坏意图的工具边界

薄弱边界	问题	更好的能力
run_shell(command)	任意执行，并且隐式依赖主机	container.image.scan(imageRef)
run_az_cli(args)	提供商范围的命令界面与环境凭据	cloud.resource.inventory(scope, filters)
call_rest(method, url, body)	模型选择端点、版本与安全敏感标头	network.privateAccess.audit(resourceId)
execute_sql(query)	无界数据访问，难以进行语义验证	finops.cost.read(scope, period, grouping)
manage_vm(action, id, options)	读取、调整大小、重启与删除压缩为一种权限	compute.vm.rightsize.plan(resourceId, targetSku)
manage_knowledge(action, payload)	摄取、删除、查询与别名变更共用一个界面	rag.knowledge.refresh(knowledgeBaseId, sourceVersion)

20.4 分离观测、建议、规划、执行、验证与回滚

安全智能体不会从自然语言直接跳到修改，而是沿操作阶梯前进。每一阶都有不同证据要求，并可使用不同身份。观测读取当前状态；建议解释证据；规划解析确切目标并计算拟议变更；执行应用已批准计划；验证独立读取结果状态；当变更失败或产生不可接受结果时，回滚恢复已知安全状态。

这种分解在云控制平面中尤其有价值，因为同一用户请求可能映射到多项提供商操作。“降低存储成本”可能意味着更改生命周期规则、删除孤立磁盘、把对象迁移到更冷层，或缩短保留期。建议可以保持定性，但计划必须识别确切资源 ID、当前值、目标值、预期成本效果、风险与回滚路径。



图 20-3 能力从观测走向产生效果时，权限随之增加。

表 20-4 操作阶梯契约

阶段	典型输出	权限与证据
观测	规范化当前状态快照	只读身份；来源时间戳与提供商查询证据。
建议	带理由与不确定性的候选操作	不修改；引用证据并估算影响。
规划	确切目标集、差异、前置条件、成本估算、回滚、摘要	以确定性方式解析当前状态；必须进行策略检查。
执行	提供商操作 ID 与即时响应	狭窄修改身份；已批准不可变计划；幂等键。
验证	独立的变更后读取与结果指标	优先使用独立读取路径；对照计划与 SLO 比较。
回滚	恢复的状态或补偿操作	预先声明的回滚权限；验证恢复与残余风险。

关键边界：批准位于规划与执行之间。批准者应看到已解析目标与效果，而不只是自然语言请求。

20.5 将身份与权限绑定到操作

智能体工具经常跨越多个身份域。用户或上游服务发起请求，应用验证调用方，智能体平台选择能力，工具代理可以使用工作负载身份、交换委派令牌或调用提供商管理连接，目标资源执行最终授权检查。这些身份不能压缩成一把 API 密钥。

两种模式反复出现。在委派模式中，工具在调用方现有权限内行动，适合下游系统已有有意义的每用户授权场景，例如文档存储库或企业应用。在工作负载模式中，限定作用域的服务身份执行一项组织操作，例如在已批准资源组作用域写入预算。工作负载模式更易自

动化，却会把问责转移给应用所有者。

能力契约必须独立于模型识别目标作用域。资源 ID、订阅、项目、资源组、区域、工作区或租户边界，应从经过身份验证的应用上下文或允许列表目录中解析。自由格式模型文本不是可信权限来源。凭据绝不能嵌入提示词、工具描述、检索文档或返回输出。

MCP 与 OpenAPI 可以描述身份验证机制，托管平台可以集中连接，但传输层授权并不消除最小权限设计。OAuth 安全指南强调限定作用域的令牌与安全流程；MCP 授权则基于保护工具与用户特定数据的标准化授权模式。[6][8][9]

表 20-5 受治理工具调用的身份链

身份或决策	问题	所需证据
终端用户或调用方	谁请求了结果？	已验证身份的主体、组织或租户、同意上下文。
应用	哪个产品界面接受了请求？	应用 ID、端点、发布、相关 ID。
智能体	哪个智能体版本选择了能力？	智能体发布摘要与工具选择追踪。
工具代理	哪个策略准入或拒绝了调用？	能力版本、策略决策、风险类别、批准 ID。
执行主体	哪个身份到达目标服务？	主体 ID、令牌受众或连接引用、角色作用域。
目标服务	授权并应用了什么操作？	提供商审计或操作 ID、目标资源 ID、结果。

20.6 架构约束形态；验证器约束含义

函数调用、OpenAPI 与 MCP 都依赖机器可读架构。JSON Schema 提供结构验证的共同词汇；OpenAPI 又加入 HTTP 操作、参数、媒体类型与安全架构；MCP 通过名称、元数据和架构公开工具，让客户端可以发现和调用。标准降低集成阻力，但语法有效的请求仍可能不安全或不合理。[3][4][5]

因此，验证应分层进行。结构验证检查类型、必填字段、格式、枚举与边界；引用验证确认资源 ID 存在并属于允许作用域；语义验证检查业务规则，例如预算周期是否允许、目标 SKU 是否支持工作负载、存储层变化是否符合保留策略，或知识源版本是否获准；状态验证检查前置条件与并发；策略验证应用组织规则；成本验证拒绝超出允许成本或节省包络的操作。

描述应面向模型选择编写，验证器应面向对抗性输入编写。有帮助的描述会说明何时调用工具以及工具不做什么，而不试图在自然语言中编码授权。恶意提示词可以要求模型忽略自然语言，却无法让目标资源授予执行主体并不拥有的角色。

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "finops.budget.create input",
  "type": "object",
  "additionalProperties": false,
  "required": ["scopeId", "amount", "timeGrain", "notificationProfile"],
  "properties": {
    "scopeId": {"type": "string", "pattern":
"^/subscriptions/[^/]+/resourceGroups/[^/]+$"},
    "amount": {"type": "number", "exclusiveMinimum": 0, "maximum": 1000000},
    "timeGrain": {"enum": ["Monthly", "Quarterly", "Annually"]},
    "notificationProfile": {"type": "string", "enum": ["team-default", "finance-
review"]},
    "idempotencyKey": {"type": "string", "minLength": 16, "maxLength": 128}
  }
}
```

示例 20-2：有界预算能力的结构架构

表 20-6 模型生成后的验证层

层	问题示例	失败响应
结构	字段是否存在、类型正确、在边界内且没有未知属性？	在调用任何提供商操作前拒绝。
引用	目标是否存在并属于经过身份验证的作用域？	返回不披露信息的目标错误。
语义	请求的状态对该能力是否合理？	返回可操作的验证详情。
状态与并发	资源自计划创建后是否发生变化？	拒绝过期计划；重新构建并批准。
策略	是否允许该区域、标签、所有者、数据类别或操作？	记录策略拒绝及所需修复。
成本与配额	操作是否超出预算、配额或单位经济性约束？	拒绝、缩小作用域或要求更高级批准。

20.7 风险类别、批准与人工控制

工具治理应对成功执行的后果进行分类，而不只是输入敏感度。只读清单查询可能公开敏感元数据，却不改变状态；预算创建调用会更改控制平面资源与通知行为；VM 调整大小可能可逆，却影响服务；删除密钥、更改身份策略、发送外部消息或承诺采购，可能产生不可逆或具有法律意义的效果。

OWASP 把过度自主描述为过度功能、过度权限或过度自主性。缓解措施不是更好的系统提示词，而是更窄的工具界面、最小权限执行、独立验证，以及高影响操作的明确批准。[12][13]

批准必须绑定确切计划。批准记录应包括能力版本、目标资源 ID、参数值、预期效果、计划摘要、批准者、有效期与所有条件。如果当前状态变化或计划重新构建，摘要就会变化，批准也不再适用。模型不得批准自己的高影响操作；用户先前说过“优化一切”之类的话，也不足以授权无界的未来变更。

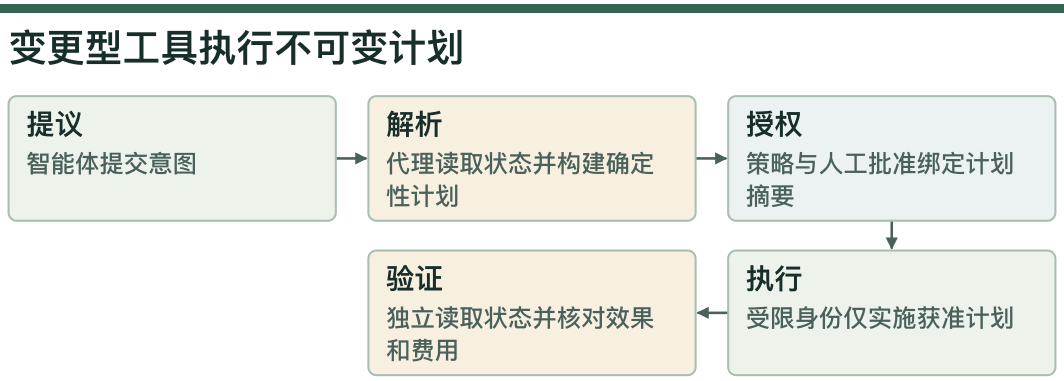


图 20-4 修改能力应执行不可变、经过策略检查且已批准的计划。

表 20-7 能力风险类别示例

类别	典型能力	默认控制
R0——信息型	读取公共文档或非敏感运行状况	自动执行；正常遥测。
R1——限定作用域读取	盘点资源、读取成本、检查配置	最小权限身份；数据最小化；速率限制。
R2——有界可逆变更	创建预算、更改标签、提出 GitOps 变更	试运行或规划；策略检查；按作用域批准。
R3——影响服务或成本高昂的变更	调整计算大小、更改网络暴露、轮换部署	明确批准计划；金丝雀或维护窗口；回滚。
R4——破坏性、权限、财务或外部效果	删除数据、授予角色、购买容量、发送外部通信	确定性工作流；强重新身份验证；职责分离；通常禁止智能体直接执行。

20.8 幂等性、并发与长时间运行操作

智能体会重试，网络会故障，工具服务器会超时，模型也可能两次发出同一调用。生产工具必须假设会重复交付且完成状态可能不明确。读取操作天然可重复，但修改操作需要明确幂等语义。幂等键应识别一项逻辑请求，重复到达时返回现有操作或结果。

并发保护同样重要。计划针对特定状态有效。如果另一个操作人员在执行前更改资源，工具应使用资源版本、ETag、世代或提供商特定前置条件拒绝过期计划。静默应用旧计划可能撤销合法变更，或产生与批准者审查内容不同的效果。

长时间运行的云操作不应隐藏在无限保持打开的工具调用之后。工具可以验证并接受请求，返回操作资源，公开状态与取消，并异步完成。异步请求—回复模式通常使用 HTTP 202、状态位置、进度状态与幂等键，以避免重复工作。[20][21]

表 20-8 工具契约应包含的可靠性语义

关注点	契约字段或行为	证据
重复请求	必填幂等键或天然幂等方法	重复请求返回同一操作 ID。
并发变更	ETag、世代、预期版本或计划摘要前置条件	提供商冲突或过期计划拒绝。
长时间运行操作	已接受状态、操作 URL、轮询或事件完成、取消	操作状态转换与最终提供商结果。
部分失败	步骤级状态与补偿操作	已完成步骤、残余资源、回滚状态。
超时	已定义客户端与服务器截止时间	超时分类；区分未知与失败。
重试	基于错误类别的有界重试策略	尝试次数、退避、最终状态。

20.9 结果信封与不可信工具输出

不应把提供商响应作为不透明字符串插入下一轮模型对话。工具适配器应将其规范化为稳定结果信封，分离事实、效果、证据、警告与面向模型的解释。这样，应用可以直接呈现权威字段，同时只向模型提供继续推理所需信息。

工具输出是不可信输入。外部 API、检索文档、工单描述、日志消息或浏览器页面可能包含试图重定向智能体的文本。工具边界应把内容标记为数据，保留来源，应用大小与数据丢失控制，并避免让原始输出重新定义智能体指令或权限。模型可以总结结果，但不得把结果字符串转化为许可。

```
{
  "capabilityId": "finops.budget.create",
  "capabilityVersion": "1.0.0",
  "requestId": "req_01J...",
  "operationId": "op_01J...",
  "status": "succeeded",
```

```
  "target": {"provider": "azure", "resourceId":
"/subscriptions/.../budgets/team-monthly"},

  "effect": {"created": true, "amount": 5000, "currency": "USD", "timeGrain":
"Monthly"},

  "verification": {"method": "provider-read", "observedAt": "2026-08-
25T09:15:00Z"},

  "evidence": {"traceId": "...", "policyDecisionId": "...",
"providerOperationId": "..."},

  "cost": {"toolExecution": 0.0012, "currency": "USD"},

  "warnings": [],

  "rollback": {"supported": true, "capabilityId": "finops.budget.delete"}

}
```

示例 20-3：稳定工具结果信封

表 20-9 结果信封字段

字段	用途
能力与版本	把行为绑定到经过审查的契约。
请求与操作 ID	关联重试、异步工作与提供商证据。
状态	区分已接受、运行中、成功、失败、拒绝、已取消与未知。
目标	识别实际提供商与资源边界，但不暴露秘密。
效果	说明发生了什么变化，而不只是提供商返回了什么。
验证	记录如何以及何时独立观测到效果。
证据	关联追踪、策略、批准、提供商操作与审计记录。
成本与警告	支持 FinOps、安全和面向用户的决策。
回滚	识别是否支持逆转以及由哪项能力执行。

20.10 工具的可观测性、审计与 FinOps

工具调用应参与可观测性章节所述的同一托管反馈回路。追踪应连接用户请求、智能体发布、工具选择、策略结果、批准、执行主体、提供商操作、验证读取与最终答案。追踪不得不加区分地记录提示词、凭据或敏感提供商负载；证据设计包括脱敏与保留策略。

运营指标应针对能力。有用度量包括调用率、成功率与拒绝率、验证失败、策略拒绝、批准延迟、执行延迟、提供商限流、过期计划冲突、回滚率、验证失败和每个成功结果的成本。模型级任务完成分数很高，也可能掩盖一个经常违反策略或需要人工修复的工具。

成本归因应分离模型选择、工具执行、提供商消费与下游业务效果。成本读取工具本身可能很便宜，其建议的操作却可能产生大额承诺。因此，能力卡应区分执行成本、预期资源成本变化，以及验证后的实际节省或价值。



图 20-5 证据包连接请求、决策、计划、执行、验证与审计。

表 20-10 智能体工具的最低运营信号

信号家族	示例	支持的决策
需求	调用、唯一调用方、工具选择率	容量、目录相关性、弃用。
质量	验证失败、错误目标防止、验证不匹配	架构改进与安全审查。
可靠性	延迟、超时、重试、提供商错误、回滚	SLO、韧性、向提供商升级。
授权	被拒绝调用、作用域不匹配、令牌或连接故障	最小权限审查与事件检测。

信号家族	示例	支持的决策
人工控制	批准率、批准延迟、被拒绝计划、过期批准	风险类别调优与工作流设计。
成本与价值	执行成本、提供商成本差异、实际节省或结果	FinOps 与单位经济性。

20.11 工具公开模式：函数调用、OpenAPI、MCP 与工作流

能力契约独立于公开协议。团队应根据所有权、复用、延迟、授权与运营责任选择模式。

应用拥有工具实现时，函数调用是最简单的模式。模型发出结构化请求，应用进行验证，执行本地代码或内部服务，再返回结果。它提供强控制，但要求每个应用自行管理发现、凭据与生命周期。

已有稳定 HTTP API 时，OpenAPI 很合适。OpenAPI 3.1 可以描述操作、架构、服务器、媒体类型与安全要求。面向模型的工具仍需要整理后的操作描述、有界操作与目标服务授权。庞大的企业 API 很少应整体导入。

MCP 标准化远程工具与资源的发现和调用，适合平台或领域团队维护的共享工具服务器。MCP 规范用名称与架构识别工具，其授权模型则为受保护操作提供标准路径。MCP 不决定企业风险类别、最小权限目标角色、批准策略或提供商特定回滚。

工作流连接器适合应保持确定性的流程。工作流可以收集批准、调用多个系统、等待异步完成、补偿故障并产生业务级证据。智能体可以选择工作流或为其提供参数，而工作流保留确切顺序和控制。当平台拥有执行环境且安全契约明确时，托管内置工具也很合适。

表 20-11　工具公开模式

模式	最佳适用场景	治理重点
函数调用	应用拥有、低延迟、限定作用域的代码	由应用执行；验证每个参数；在本地管理身份与生命周期。
OpenAPI 工具	已有稳定操作的 HTTP 服务	整理操作；声明安全；使用最小权限托管身份或连接。
MCP 服务器	跨智能体与客户端共享工具	服务器所有权、OAuth 授权、目录版本控制、批准、租户、审计。
工作流或连接器	多步业务流程、批准、等待、补偿	确定性顺序、连接治理、步骤证据、重放行为。
内置托管工具	搜索、代码执行、文件检索或提供商托管能力	平台边界、数据处理、沙箱、配额与追踪可见性。

协议规则：MCP、OpenAPI 或函数调用可以标准化调用。它们都不能替代能力所有权、提供商授权、策略、批准、验证或退役。

20.12 Azure 参考模式：Foundry 工具箱、托管身份与提供商适配器

Microsoft Foundry 提供了有用的参考模式，因为它区分内置与自定义工具，并建议使用工具箱作为可复用托管集合。当前文档把工具箱描述为集中管理凭据与策略的托管 MCP 端点，允许工具版本变化而无需修改每个智能体定义。自定义工具可以通过 MCP 或 OpenAPI 连接，函数调用则把执行留在应用中。[14][15][16]

Azure 实现可以按如下方式保留能力契约：Foundry 智能体只看到整理后的工具元数据；工具箱或应用网关验证请求身份并绑定已批准能力版本；OpenAPI 服务、Azure Function 或 Logic Apps 工作流验证参数；托管身份只获得该能力所需的目标服务角色；Azure Resource Manager 或相关数据平面 API 执行操作；Azure Monitor、Application Insights、Activity Log 与领域特定日志记录证据。

当目标服务支持 Microsoft Entra 身份验证时，托管身份更可取，因为运行时不存储可复用秘密。Foundry OpenAPI 工具目前支持匿名、API 密钥与托管身份验证；托管身份仍需要目标资源作用域内的最小权限角色。OpenAPI operationId 应描述清楚且保持稳定，规范只应公开供模型使用的有界操作。[15]



图 20-6 共享能力目录保留工程意图，提供商适配器保留不等价性。

表 20-12 按契约层划分的 Azure 参考实现

契约层	Azure 实现选项
发现与版本控制	Foundry 工具箱或应用目录；以存储库为基础的能力卡与 OpenAPI 规范。
身份验证	Foundry 项目连接、Microsoft Entra 委派流或托管身份。
执行适配器	Azure Functions、App Service 或 Container Apps API、Logic Apps 工作流或内部服务。
目标授权	最窄作用域的 Azure RBAC 与服务特定数据平面角色。
策略与批准	Azure Policy、应用策略引擎、工作流批准、部署门禁。
证据	Application Insights 或 OpenTelemetry 追踪、Azure Monitor 指标与日志、Activity Log、提供商操作 ID。
生命周期	工具箱或 API 版本、IaC 部署、发布清单、弃用与身份撤销。

20.13 多云转换：保留意图，记录不等价性

多云能力目录不应假装所有提供商都实现相同资源模型。稳定能力表达意图；每个适配器记录该意图如何映射到提供商对象、授权、异步行为、错误、可观测性、成本与删除。如果一家提供商无法保留某项必要不变量，适配器应报告不等价，而不是模拟成功。

阿里云 Model Studio 支持 MCP 与自定义插件；阿里云 AgentRun 描述了支持 MCP 和函数调用的 Tool Hub，提供统一工具调用、凭据管理、可观测性与无服务器执行。腾讯云 ADP 在其应用模式内公开基于插件和 MCP 的工具，并提供工作区与应用权限模型。火山引擎 AgentKit 公开 MCP 服务与工具集管理，并可把已有 REST 或 OpenAPI 服务适配为面向 MCP 的工具界面。[24][25][26][27][28][29][30]

工程比较应聚焦六个问题：哪个对象表示工具或插件？如何验证身份？凭据存在哪里？如何表达用户与工作区权限？如何观测并计费执行？如何禁用或退役工具版本？产品名称和界面组织各异，这些运营问题保持稳定。

表 20-13 多云工具集成映射

工程意图	Azure	阿里云	腾讯云	火山引擎
共享工具目录	Foundry 工具箱 / MCP 端点	Model Studio 插件；AgentRun Tool Hub	ADP 插件市场与 MCP 工具	AgentKit MCP 服务与工具集
HTTP API 适配	Foundry OpenAPI 工具	自定义插件或 API 支持的 MCP / 函数调用	通过 API 与代码实现的自定义插件	把 REST / OpenAPI 适配到 MCP 网关

工程意图	Azure	阿里云	腾讯云	火山引擎
工作负载凭据	托管身份或项目连接	RAM 角色、服务关联角色、凭据管理	CAM 角色及平台或应用权限	IAM 与 AgentKit 身份验证配置
确定性工作流	Logic Apps 或 Functions 工作流	Model Studio 工作流 / Function Compute / AgentRun	ADP 工作流模式	AgentKit 工作流与运行时集成
证据与运维	OpenTelemetry、Application Insights、Azure Monitor、Activity Log	AgentRun 追踪、日志、ARMS 或 SLS 集成	ADP 平台日志与应用控制	AgentKit 日志、MCP 诊断、平台监控

20.14 能力目录是一项产品

工具目录不是一个装 JSON 文件的文件夹，而是一项拥有所有者、使用者、服务目标、安全审查、文档、兼容策略、支持渠道和退役责任的内部产品。目录应公开少量高质量能力，而不是每一项提供商操作。

每项能力都经历生命周期：草稿、已验证、已批准、已发布、已弃用、已退役。草稿能力可以在隔离环境中测试；验证确认架构、授权、失败行为与证据；批准记录所有权与风险接受；发布让获准智能体能够发现能力；弃用阻止新绑定并识别替代项；退役禁用调用、撤销凭据、归档所需证据，并验证没有活动智能体发布仍依赖该能力。

版本控制应反映行为，而不是文件编辑。当权限、副作用、语义或必填输入发生不兼容变化时，应提升主版本；新增可选字段或提供商映射而不改变既有含义时，可以提升次版本；修正文档或兼容适配器缺陷时，可以提升修订版本。工具描述也会影响模型选择，应有意发布；“只改措辞”也可能改变行为，因此值得评估。

表 20-14 能力生命周期门禁

状态	准入标准	退出证据
草稿	具名所有者、用途、初步架构、提供商原型	原型测试与已知限制。
已验证	通过架构、语义、授权、幂等性、失败与证据测试	验证报告与威胁审查。
已批准	风险所有者接受策略、作用域、身份、成本与支持模型	批准记录与发布摘要。
已发布	已批准智能体可以发现，且具有已记录的 SLO 与版本	目录项、监控、支持联系人。
已弃用	存在替代或退役路径；阻止新绑定	依赖清单与迁移通知。

状态	准入标准	退出证据
已退役	没有活动绑定；端点已禁用；身份与凭据已撤销	调用失败测试与已归档证据。

能力检查 C6：编写并审查五张云能力卡

C6.1 能力目标

从本书其他章节已经验证的云能力中创建一个小型、与提供商无关的能力目录。目录至少包含一项只读能力与一项修改能力。每张卡必须足够精确，使独立审查者无需阅读适配器源代码，就能判断身份、作用域、副作用、验证、证据、成本边界与生命周期。

C6.2 建议能力集

表 20-15 建议的五卡能力集

能力 ID	模式	主要来源章节	用途
cloud.resource.inventory	只读	云控制平面与 IaC	返回已批准作用域的规范化清单。
network.exposure.audit	只读分析	网络与私有连接	识别公共暴露与缺失的私有访问控制。
finops.budget.create	修改	作为云控制平面的 FinOps	根据已批准计划创建限定作用域的预算与通知策略。
gitops.change.propose	修改，但属间接修改	持续交付与变更治理	创建可审查的配置变更，而不直接修补生产。
rag.knowledge.refresh	异步修改	作为托管知识资源的 RAG	刷新已批准知识源，并验证已发布别名或版本。

C6.3 创建目录结构

为每项能力创建一个目录。存储 capability.yaml、input.schema.json、output.schema.json、provider-mapping.md、test-cases/ 与 evidence-example.json。秘密与实时凭据应存放在目录之外。

使用描述意图而非提供商产品名称的稳定能力 ID。分别记录当前提供商适配器，使提供商改名时无需强制更改每个智能体绑定。

C6.4 完成每张能力卡

每张卡都要指定用途、非目标、所有者、版本、生命周期状态、模式、风险类别、架构、目标作用域规则、身份路径、所需目标操作、验证层、批准规则、幂等行为、超时、验证方法、回滚、证据字段、成本边界、数据分类、租户假设与提供商映射。

修改卡必须包含试运行或确定性规划步骤，以及绑定不可变计划摘要的批准规则。只读卡仍须定义数据最小化、速率限制和允许资源作用域。

C6.5 审查模型可见元数据

编写能帮助模型正确选择的简洁工具名称与描述，包括不应使用该工具的情形。不要把凭据、隐藏授权规则或提供商特定命令语法放入描述。

测试模糊请求。例如，确认“显示资源”会选择 cloud.resource.inventory，但“删除未使用资源”不会；确认用户只要求建议时，不会选择 finops.budget.create。

C6.6 审查权限与风险

绘制从调用方到目标服务的身份链。确认执行主体只拥有能力所需操作。验证目标作用域来自经过身份验证的上下文或允许列表标识符，而不是自由格式模型文本。

根据表 20-7 分配风险类别。记录策略能否自动批准、是否需要人工、是否需要重新身份验证，以及是否禁止智能体直接执行。

C6.7 审查可靠性与证据

定义重复交付行为、并发前置条件、长时间运行操作处理、取消、重试、超时语义与部分失败行为。分别提供一个成功结果信封和一个拒绝或失败结果信封示例。

列出构成证据包的追踪跨度、策略决策、批准、提供商操作 ID、验证读取、成本字段与保留期。

C6.8 映射四个云平台

对每项能力，映射 Azure、阿里云、腾讯云和火山引擎的提供商对象、API、身份机制、可观测证据与清理。明确标记不等价性。一家提供商具有实质不同语义时，不要强求共同字段。

C6.9 对抗性审查问题

- 不可信文档或工具输出能否让模型选择用户没有请求的修改能力？
- 调用方能否替换来自其他订阅、项目、工作区或租户的资源 ID？
- 同一请求重放后能否产生重复效果？
- 计划或目标状态变化后，过期批准能否复用？

- 执行主体能否执行能力未表示的操作?
- 工具能否在没有独立验证读取的情况下声称成功?
- 提供商输出能否向智能体对话注入新指令?
- 已弃用能力能否继续绑定到活动智能体发布?

C6.10 验证标准

- 恰好存在五张能力卡，并且至少一张只读、一张修改。
- 每张卡都命名所有者、版本、生命周期状态、模式、风险类别、身份路径、目标作用域、架构与证据契约。
- 修改能力具有规划 / 试运行边界、批准规则、幂等行为、验证，以及回滚或补偿声明。
- 所有卡均包含 Azure、阿里云、腾讯云与火山引擎映射，并记录不等价性。
- 审查者能够根据能力卡与测试用例复现允许路径与拒绝路径。
- 目录中没有凭据、秘密、实时令牌或不受限制的提供商命令。

C6.11 成本与清理边界

能力卡练习无需创建计费云资源即可完成。若执行实时适配器测试，请使用相关来源章节的专用实验作用域与清理契约。只保留经过脱敏的证据示例、架构文件、审查结果与提供商映射。测试后撤销临时身份与项目连接。

20.15 CloudDesk 移交：能力基础，而非运行时

配套书中介绍的 CloudDesk FinOps Agent 可以使用该目录，但不应重新定义能力。桌面与无服务器运行时将提供租户解析、会话与响应绑定、批准交互、任务编排、工具执行与跨租户隔离；目录提供这些机制所执行的稳定契约。

这种分工很重要。运行时可能安全，却仍公开不安全能力；能力可能设计良好，却由一个丢失租户上下文或误路由响应的运行时执行。第一本书建立安全云语义，第二本书建立跨桌面、网关、无服务器工作进程与云服务保留这些语义的运行时。

表 20-16 初始 CloudDesk 能力移交

能力	CloudDesk 用途	配套书要求的运行时行为
cloud.resource.inventory	构建限定租户作用域的云清单与所有权视图	解析租户云账户；绑定作用域；只缓存并返回租户合格结果。
network.exposure.audit	检测公共端点与缺失私有访问	运行只读适配器；保留证据；把发现路由到正确租户工作区。
finops.budget.create	在建议后创建已批准预算	生成计划；桌面批准；使用狭窄身份执行；验证并记录效果。

能力	CloudDesk 用途	配套书要求的运行时行为
gitops.change.propose	把建议转化为可审查配置变更	创建绑定租户的分支或拉取请求；绝不绕过存储库策略。
rag.knowledge.refresh	刷新租户成本策略与历史报告	按租户与工作区隔离来源、队列、索引、别名、追踪与结果。

边界提醒：本章定义能力契约。租户数据模型、跨租户隔离、流式响应绑定、队列分区、桌面批准协议与自定义无服务器工具执行属于《无服务器 AI 智能体开发》。

20.16 要点

- 只有当用途、权限、验证、风险、证据、成本与生命周期都明确后，云 API 才成为智能体工具。
- 工具描述与架构影响模型选择；工具网关与目标服务执行授权。
- 意图级工具比原始 CLI、Shell、SQL、SDK 或通用 REST 访问更安全、更可移植。
- 观测、建议、规划、执行、验证与回滚是不同权限阶段，不应压缩。
- 修改工具应执行不可变、经过策略检查、已批准且具有幂等与并发控制的计划。
- 工具结果是不可信数据，应规范化为带来源与验证的稳定结果信封。
- 函数调用、OpenAPI、MCP、工作流与内置工具是公开模式，不能替代治理。
- 多云能力目录保留工程意图，提供商适配器保留真实差异。
- 能力目录是一项产品，具备所有权、SLO、安全审查、版本控制、弃用与退役。
- CloudDesk 可以建立在这些契约上，但其租户系统与无服务器运行时必须独立保留隔离与执行正确性。

参考文献

[1] 美国国家标准与技术研究院：《零信任架构》（SP 800-207）。https://doi.org/10.6028/NIST.SP.800-207 访问于 2026 年 8 月 25 日。

[2] 美国国家标准与技术研究院：《人工智能风险管理框架（AI RMF 1.0）》。https://doi.org/10.6028/NIST.AI.100-1 访问于 2026 年 8 月 25 日。

[3] OpenAPI Initiative：OpenAPI 规范 3.1.1。https://spec.openapis.org/oas/v3.1.1.html 访问于 2026 年 8 月 25 日。

[4] JSON Schema：JSON Schema 规范，草案 2020-12。https://json-schema.org/specification 访问于 2026 年 8 月 25 日。

[5] Model Context Protocol：工具，规范 2026-07-28。https://modelcontextprotocol.io/specification/2026-07-28/server/tools 访问于 2026 年 8 月 25 日。

[6] Model Context Protocol：理解 MCP 中的授权。https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/authorization 访问于 2026 年 8 月 25 日。

[7] Model Context Protocol：规范 2026-07-28。https://modelcontextprotocol.io/specification/2026-07-28 访问于 2026 年 8 月 25 日。

[8] IETF：RFC 9700——OAuth 2.0 安全当前最佳实践。https://www.rfc-editor.org/info/rfc9700/ 访问于 2026 年 8 月 25 日。

[9] IETF：RFC 9728——OAuth 2.0 受保护资源元数据。https://www.rfc-editor.org/info/rfc9728/ 访问于 2026 年 8 月 25 日。

[10] 云原生计算基金会：CloudEvents 规范。https://cloudevents.io/ 访问于 2026 年 8 月 25 日。

[11] OpenTelemetry：生成式 AI 系统语义约定。https://opentelemetry.io/docs/specs/semconv/gen-ai/ 访问于 2026 年 8 月 25 日。

[12] OWASP Gen AI 安全项目：LLM06:2025 过度自主。https://genai.owasp.org/llmrisk/llm062025-excessive-agency/ 访问于 2026 年 8 月 25 日。

[13] OWASP Cheat Sheet Series：AI 智能体安全速查表。https://cheatsheetseries.owasp.org/cheatsheets/AI_Agent_Security_Cheat_Sheet.html 访问于 2026 年 8 月 25 日。

[14] Microsoft Learn：Microsoft Foundry Agent Service 中的工具类型。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/tool-catalog 访问于 2026 年 8 月 25 日。

[15] Microsoft Learn：将 OpenAPI 工具连接到 Microsoft Foundry Agents。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/tools/openapi 访问于 2026 年 8 月 25 日。

[16] Microsoft Learn：将智能体连接到 MCP 服务器端点。https://learn.microsoft.com/en-us/azure/foundry/agents/how-to/tools/model-context-protocol 访问于 2026 年 8 月 25 日。

[17] Microsoft Learn：Microsoft Foundry Agent Service 工具最佳实践。https://learn.microsoft.com/en-us/azure/foundry/agents/concepts/tool-best-practice 访问于 2026 年 8 月 25 日。

[18] Microsoft Learn：使用托管身份验证 Azure Logic Apps 连接。https://learn.microsoft.com/en-us/azure/logic-apps/authenticate-with-managed-identity 访问于 2026 年 8 月 25 日。

[19] Microsoft Learn：Web API 设计最佳实践。https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design 访问于 2026 年 8 月 25 日。

[20] Microsoft Learn：异步请求—回复模式。https://learn.microsoft.com/en-us/azure/architecture/patterns/asynchronous-request-reply 访问于 2026 年 8 月 25 日。

[21] Microsoft Learn：Web API 实现最佳实践。https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-implementation 访问于 2026 年 8 月 25 日。

[22] Microsoft Learn：Azure Resource Manager 模板概览。https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/overview 访问于 2026 年 8 月 25 日。

[23] Microsoft Learn：Azure Logic Apps 概览。https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview 访问于 2026 年 8 月 25 日。

[24] 阿里云文档：Model Studio MCP。https://www.alibabacloud.com/help/en/model-studio/mcp 访问于 2026 年 8 月 25 日。

[25] 阿里云文档：Model Studio 自定义插件。https://www.alibabacloud.com/help/en/model-studio/custom-plug-ins 访问于 2026 年 8 月 25 日。

[26] 阿里云文档：《什么是 AgentRun》。https://www.alibabacloud.com/help/en/agentrun/what-is-agentrun 访问于 2026 年 8 月 25 日。

[27] 阿里云文档：在 Model Studio 高代码应用中使用智能体身份。https://www.alibabacloud.com/help/en/agentidentity/using-agent-identity-in-bailian-high-code 访问于 2026 年 8 月 25 日。

[28] 腾讯云文档：智能体开发平台插件介绍。https://intl.cloud.tencent.com/document/product/1254/72577 访问于 2026 年 8 月 25 日。

[29] 腾讯云文档：基于 API 与代码创建智能体应用。https://intl.cloud.tencent.com/document/product/1254/72579 访问于 2026 年 8 月 25 日。

[30] 腾讯云文档：平台侧用户权限。https://intl.cloud.tencent.com/document/product/1254/73347 访问于 2026 年 8 月 25 日。

[31] 腾讯云文档：应用侧用户权限说明。https://intl.cloud.tencent.com/document/product/1254/72504 访问于 2026 年 8 月 25 日。

[32] 火山引擎文档：AgentKit MCP 服务管理。https://docs.volcengine.com/docs/86681/1913814?lang=en 访问于 2026 年 8 月 25 日。

[33] 火山引擎文档：AgentKit 代码沙箱工具示例。https://docs.volcengine.com/docs/86681/1860267?lang=en 访问于 2026 年 8 月 25 日。

[34] 火山引擎文档：AgentKit A2A 与 MCP 工具集成。https://www.volcengine.com/docs/86681/2249666 访问于 2026 年 8 月 25 日。

[35] FinOps 基金会：FinOps 框架。https://www.finops.org/framework/ 访问于 2026 年 8 月 25 日。

[36] FinOps 基金会：FOCUS 规范。https://focus.finops.org/ 访问于 2026 年 8 月 25 日。

[37] Google：《网站可靠性工程工作簿——实施 SLO》。https://sre.google/workbook/implementing-slos/ 访问于 2026 年 8 月 25 日。

[38] CNCF：OpenTelemetry 文档。https://opentelemetry.io/docs/ 访问于 2026 年 8 月 25 日。

主题索引

A–C

条目按拼音或英文首字母排列。§ 指向讲解小节，“实作”指向实验或练习。 中点后为本版 PDF 页码。

A 阿里云 § 3.6 · 46 § 16.8 · 429

AI PaaS（AI 平台即服务） § 16.5 · 425 § 17.2 · 440

AIOps（智能运维） § 13.7 · 347

安全态势管理（CSPM） § 15.1 · 392 § 15.8 · 400

API 管理 § 9.7 · 213 实作 9.12 · 219

Argo CD § 12.6 · 313

Azure 参考控制平面 § 1.6 · 9 § 13.4 · 342 § 17.13 · 451

B 版本晋级 § 10.3 · 239 § 12.3 · 310 实作 18.9.12 · 487

保留策略 § 8.6 · 180 § 13.10 · 350 实作 8.14 · 188

备份与恢复 § 6.6 · 121 § 8.5 · 179

标签与成本归属 § 4.3 · 60 § 4.4 · 62 实作 4.12 · 72

Bicep § 4.7 · 66 实作 14.12.4 · 383

C 策略即代码 § 15.4 · 396 实作 15.15 · 408

成本分摊 § 4.4 · 62 § 14.4 · 369

成本异常与预测 § 14.5 · 370

持久卷声明（PVC） § 11.9 · 279 实作 11.14.7 · 294

持久状态 § 11.9 · 279 图 11-9 · 286 实作 11.14.7 · 294

持续交付 § 12.2 · 309 § 12.10 · 318

持续验证 § 13.8 · 348 § 15.6 · 398

出口路由 § 7.4 · 146

索引（续）

C–G

C 处理边界（模型） § 17.7 · 445

词法检索 § 18.5.1 · 473 § 18.5.3 · 473

CloudDesk § 4.14 · 80 § 14.13 · 387 § 20.15 · 547

ConfigMap 与 Secret § 11.8 · 278 实作 11.14.4 · 287

错误预算 § 13.5 · 344

错误预算消耗率 § 13.5.3 · 346

D 单位经济性 § 14.7 · 373 § 17.12 · 450 § 19.14 · 511

弹性伸缩 § 6.4 · 118 § 11.10 · 280

倒数排名融合（RRF） § 18.5.3 · 473 实作 18.9.7 · 484

DNS（域名系统） § 7.3 · 146 § 7.8 · 150 实作 7.14.7 · 164

多云等价性 § 3.7 · 47 § 3.8 · 49 § 20.13 · 543

F 发布回滚 § 9.4 · 207 § 12.11 · 319 实作 12.15.7 · 330

FinOps（云成本管理） § 14.2 · 366 § 14.10 · 379 实作 14.12 · 382

Flux § 12.6 · 313 实作 12.15.4 · 327

服务发现 § 11.7 · 277 实作 11.14.6 · 293

服务级别目标 见 SLO（服务级别目标）

服务级别指标（SLI） § 13.5 · 344 实作 13.12.8 · 357

服务主体 § 5.4.1 · 93

G 告警 § 13.6 · 346 实作 13.12.10 · 358

GitOps § 12.5 · 312 实作 12.15.5 · 328

索引（续）

G–J

G 工程意图 § 3.7 · 47 § 7.13 · 155 § 20.13 · 543

公共暴露 § 7.5 · 147 实作 7.14.6 · 163

工具风险与审批 § 20.5 · 533 § 20.7 · 536

工具粒度 § 20.3 · 531

工具目录 § 20.14 · 544 实作 C6.4 · 546

工具契约 § 19.8 · 503 § 20.2 · 528

工具输出 § 20.9 · 538

供应链证据 § 10.7 · 245 实作 10.14.9 · 260

工作负载控制器 § 11.5 · 274 图 11-4 · 273

工作负载身份 § 5.3 · 91 § 5.4.4 · 95 § 17.6 · 444

工作负载形态 § 6.2 · 116 § 6.3 · 117

故障域 § 6.5 · 120 § 6.13 · 136

观测信号 § 13.2 · 338 § 13.3 · 340

H 函数计算 § 9.5 · 208 实作 9.13 · 223

合规证据 § 15.7 · 399 实作 15.15 · 408

混合检索 § 18.5.3 · 473 实作 18.9.7 · 484

火山引擎 § 3.6 · 46 § 16.10 · 430

I IAM（身份与访问管理） 见 身份与访问管理（IAM）

J 基础设施代码 见 基础设施即代码（IaC）

基础设施即代码（IaC） § 4.6 · 65 § 4.10 · 70 实作 4.13 · 75

索引（续）

J–M

J 计算容量 § 6.7 · 122 § 6.8 · 123 实作 6.12 · 127

健康探针 § 11.6 · 275 实作 11.14.4 · 287

检索评估 § 18.8.1 · 478 § 18.8.2 · 479 实作 18.9.11 · 487

检索授权 § 18.3.4 · 470 § 18.6.3 · 475 实作 18.9.7 · 484

检索增强生成 见 RAG（检索增强生成）

节省验证 § 14.6 · 372 § 14.10 · 379

镜像签名 § 10.9 · 248 实作 10.14.7 · 259

镜像摘要 § 10.3 · 239 实作 10.14.4 · 257

K 可观测性 § 13.1 · 337 § 17.11 · 448 § 19.11 · 506

控制平面 § 1.3 · 4 § 2.4 · 24 § 19.6 · 501

跨云部署选址 § 3.4 · 43 § 3.9 · 49

Kubernetes 见 托管 Kubernetes

L 例外与风险接受 § 15.10 · 402

流量入口 § 7.6 · 149 § 11.7 · 277 实作 11.14.6 · 293

M **MaaS（模型即服务）** § 17.4 · 442 § 17.5 · 443

MCP（模型上下文协议） § 20.11 · 541

幂等性 § 2.10 · 31 § 20.8 · 537

密钥与机密 § 5.5 · 96 § 11.8 · 278 § 12.8 · 316

Microsoft Foundry § 16.7 · 428 § 17.13 · 451 § 19.15 · 512

MLOps、LLMOps 与 AgentOps § 16.6 · 427

索引（续）

M–R

M 模型安全护栏 § 17.10 · 448
模型版本退役 § 17.9 · 446
模型部署 § 17.3 · 441 § 17.5 · 443 § 17.15 · 453
模型即服务 见 MaaS（模型即服务）
模型配额与限流 § 17.8 · 446
模型上下文协议 见 MCP（模型上下文协议）
模型推理成本 § 14.9 · 377 § 17.12 · 450
模型推理端点 § 17.1 · 439 § 17.3 · 441
O OCI 镜像 § 10.2 · 237 实作 10.14.4 · 257
OpenTelemetry § 13.3 · 340 实作 13.12.5 · 355
P 漂移检测与纠正 § 4.8 · 67 § 12.9 · 317 实作 12.15.6 · 329
Pod 模板 § 11.6 · 275 实作 11.14.4 · 287
PVC 见 持久卷声明（PVC）
Q 期望状态 § 2.5 · 25 § 11.3 · 271 实作 12.15.3 · 325
权限分离 § 12.8 · 316 § 15.12 · 404
R RAG（检索增强生成） § 18.2 · 468 § 18.8 · 478 实作 18.9 · 479
RBAC（基于角色的访问控制） § 5.4.2 · 93 § 5.4.6 · 96 实作 5.10.4 · 105
容量与配额 § 6.7 · 122 § 17.8 · 446 § 19.14 · 511
软件物料清单 见 SBOM（软件物料清单）
软件制品仓库 § 10.6 · 242 实作 10.14.2 · 255

索引（续）

S–Y

S SBOM（软件物料清单） § 10.7.1 · 246

删除与清理 § 2.9 · 30 实作 7.14.12 · 166 实作 11.14.12 · 297

身份链（智能体） § 19.7 · 501 § 20.5 · 533

身份联合 § 5.4.4 · 95

身份与访问管理（IAM） § 5.1 · 87 § 5.4 · 93 § 19.7 · 501

审计轨迹 § 15.9 · 401 § 19.11 · 506 § 20.10 · 540

审批与人工控制 § 4.10 · 70 § 20.7 · 536

事件驱动的数据路径 § 8.9 · 183 § 9.6 · 211 实作 9.13 · 223

数据湖 § 8.7 · 181 § 8.8 · 182 § 16.3 · 423

数据平面 § 2.4 · 24 § 17.1 · 439 § 19.6 · 501

私有连接 § 7.7 · 149 § 18.6.2 · 475 实作 7.14.7 · 164

SLO（服务级别目标） § 13.5 · 344 § 17.11 · 448 实作 13.12.7 · 356

锁与变更证据 § 15.9 · 401

T 腾讯云 § 3.6 · 46 § 16.9 · 429

托管 Kubernetes § 11.1 · 268 § 11.2 · 270 实作 11.14 · 285

托管身份 § 5.4.1 · 93 实作 5.10.3 · 105

W 网络分段 § 7.3 · 146

文档级访问资格 § 18.5.4 · 473 § 18.6.3 · 475 实作 18.9.7 · 484

X 向量检索 § 18.5.2 · 473 § 18.5.3 · 473

Y 引用与来源 § 18.6.4 · 476 实作 18.9.10 · 486

索引（续）

Y–Z

Y 应用定义 § 11.4 · 273 图 11-4 · 273 图 11-9 · 286

预算护栏 § 14.5 · 370 实作 14.12.5 · 384

语义重排序 § 18.5.5 · 474

云成本管理 见 FinOps（云成本管理）

Z 责任共担 § 3.2 · 40 § 3.3 · 42 § 11.2 · 270

知识版本与回滚 § 18.3.5 · 470 实作 18.9.12 · 487 实作 18.9.13 · 488

知识库 § 18.2 · 468 § 18.3 · 469 实作 18.9.9 · 485

智能体记忆与会话 § 19.9 · 504

智能体评估 § 19.12 · 508

智能体运行轨迹 § 19.11 · 506 § 20.10 · 540

智能体资源契约 § 19.3 · 496 § 19.4 · 498

资源标识符 § 2.3 · 23 § 4.3 · 60 实作 1.9.3 · 15

资源范围层级 § 4.2 · 59 § 15.3 · 394

资源清单 § 4.5 · 64 实作 3.11 · 51

资源生命周期 § 2.7 · 28 § 2.9 · 30 实作 1.9.4 · 15

资源依赖图 § 1.4 · 6 § 2.6 · 26 实作 6.12 · 127

组织治理 § 15.2 · 393 § 15.3 · 394 § 15.12 · 404

最小权限 § 5.2 · 89 § 5.6 · 97 实作 5.10.7 · 108