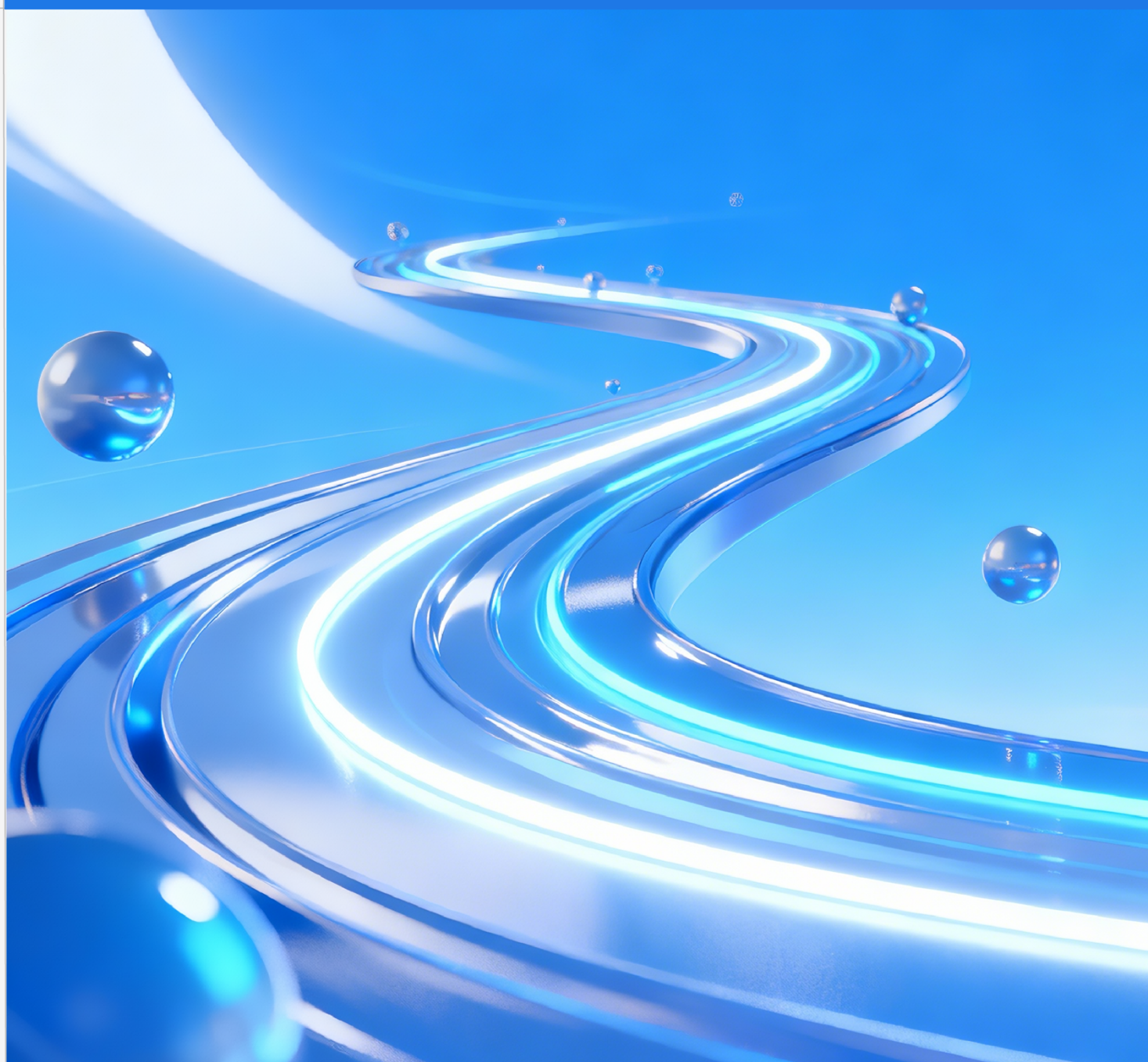
阿里云

AI 云采用框架

AI CLOUD ADOPTION FRAMEWORK



第一章．AI 云采用框架前言 5

AI 时代的新挑战 5

智能化转型进入关键阶段 5
全球智能革命不断深化，人工智能市场持续扩大 5
人工智能技术快速演进，AI 应用面临全新挑战 5

为什么需要 AI 云采用框架（AI CAF）？ 6

目标读者 6

本文件起草单位及主要起草人 7

第二章．AI 云采用框架概述 8

第三章．AI 战略 9

1. 企业 AI 愿景与战略定位 10

1.1 内外部发展环境分析 10
1.2 企业 AI 愿景确定 11
1.3 AI 战略定位与目标 12

2. AI 战略规划与关键策略 13

2.1 分阶段 AI 战略任务与路线图 13
2.2 关键策略安排 16
2.3 资源投入与配置规划 17

3.AI 应用场景选择与评估 18

3.1 AI 应用场景分类 18
3.2 基于“价值-可行性”的潜在应用场景选择 19
3.3 AI 应用场景优先级排序 20
3.4 构建“事前-事中-事后”的应用价值评估机制 21
3.5 AI 场景应用价值评估体系 23

4. AI 战略监控与迭代 27

4.1 AI 战略动态监控 27
4.2 反馈与调整机制 28

5.小结 29

第四章．AI LANDING ZONE 31

1. 资源规划 31

1.1. 概述 31
1.2. 背景与挑战 31
1.3. 具体方案 31
1.4. 采用资源组与标签来管理 AI 服务依赖的资源 37
1.5. 总结 39

2. 财务管理 40

2.1. 概述 40
2.2. 背景与挑战 40
2.3. 具体方案 40
2.4. 总结 43

3. 网络规划 43

3.1. 概述 43
3.2. 背景与挑战 43
3.3. 具体方案 44
3.4. 模型训练阶段 48
3.5. 网络总览 52
3.6. 模型推理阶段 56

4. 身份权限 65

4.1. 概述 65
4.2. 背景与挑战 65
4.3. 具体方案 65
4.4. 总结 77

5. 安全防护 78

5.1. 概述 78
5.2. 背景与挑战 78
5.3. 具体方案 79
5.4. 总结 84

6. 合规审计 85

6.1. 概述 85
6.2. 背景与挑战 85
6.3. 具体方案 85
6.4. 总结 89

7. 运维管理 90

7.1. 概述 90
7.2. 背景与挑战 90
7.3. 具体方案 91
7.4. 总结 98

8. 自动化 98

8.1. 概述 98
8.2. 背景与挑战 98
8.3. 具体方案 99
8.4. 总结 101

第五章．AI 构建 102

1. 综述 102

2. 认识 AI 智能体：定义、场景与企业挑战 103

2.1 什么是 AI 智能体 104
2.2 智能体已经在哪些行业真正用起来了 104
2.3 企业需要的智能体长什么样 105
2.4 为什么很多企业想用智能体却迟迟难以落地 106

3.智能体系统设计：范式、架构与选型指南 106

3.1 该用哪种开发范式？主流智能体开发范式解析 106
3.2 智能体系统怎么搭？单体还是多智能体 108
3.3 开发方式怎么选？低代码快速上线 vs 高代码深度定制 112
3.4 选型前必须想清楚的六个关键问题 112

4.构建智能体核心内核：规划、记忆与工具调用 117

4.1 让智能体学会思考：规划与推理机制 117
4.2 赋予智能体记忆力：从短期缓存到长期经验 119
4.3 教会智能体使用工具：连接现实世界的能力 120

5.提升输入理解能力：意图识别与查询优化 127

5.1 意图识别 127
5.2 查询重写 130

6 提示词工程：高效指令设计与系统化管理 135

6.1 提示词与提示词工程 135
6.2 怎样写出高质量提示：提示词技术 136
6.4 企业如何规模化管理提示词：构建可持续演进的提示词工程体系 142

7.知识增强：RAG 与实时信息获取 150

7.1 用检索增强生成（RAG）让回答更准、更专、更可靠 150
7.2 联网搜索：让智能体随时获取最新信息 158

8.专属模型定制：定制化后训练 160

8.1 为何需要专属模型 160
8.2 典型应用场景 161
8.3 主流训练方法和框架选择 162
8.4 训练前准备 163
8.5 训练与评估流程 166

9.数据工程：为智能体提供高质量数据燃料 170

9.1 数据工程介绍 170
9.2 企业数据的使用场景 170
9.3 企业数据接入与评估 171
9.4 数据工程核心链路 174

第六章．AI 治理 183

客户案例一：某全球运动服饰企业 AI LANDING ZONE 设计案例 185

客户案例二：某新势力汽车品牌智能驾驶训练平台 AI LANDING ZONE 实践 193

第一章．AI 云采用框架前言

AI 时代的新挑战

智能化转型进入关键阶段

全球数字化、智能化转型正处于关键跃升阶段。人工智能正加速与各行各业深度融合，推动新兴产业形态和传统产业升级。各方对人工智能应用的稳定性、安全性和可信赖性提出了更高要求，智能化发展已成为产业演进和社会进步的重要驱动力。

全球智能革命不断深化，人工智能市场持续扩大

人工智能正以前所未有的速度重构全球产业格局。全球多数企业将在生产系统中部署生成式 AI 能力，大模型驱动的认知计算正在深刻改变制造业、服务业等领域的价值链。

中国在人工智能技术研究和产业应用方面持续保持增长，相关企业数量和产业规模快速扩大，呈现出强劲的发展势头。随着行业智能化转型的深入，不同行业的差异化需求不断涌现，对 AI 算力、平台、算法模型和行业解决方案提出了更高要求。

人工智能技术快速演进，AI 应用面临全新挑战

人工智能技术进入体系化突破新阶段，推动软件工程向智能化演进。大语言模型（LLM）正在重塑软件开发模式，生成式 AI 推动人机协同开发逐渐成为主流；与此同时，对 AI 信任、风险与安全管理（TRiSM）的需求愈加迫切，模型运维（ModelOps）、智能体运维（AgentOps）、AI 安全与模型监控正成为企业关注的重点。

然而，AI 应用在大规模落地时仍面临诸多挑战：

- 数据依赖度高：高质量数据供给难度大，数据漂移可能导致模型性能退化。
- 模型迭代复杂：生命周期涵盖训练、验证、部署、监控与回滚，迭代过程对系统稳定性要求高。
- 资源需求波动大：训练阶段计算资源消耗巨大，推理阶段需低延迟与稳定性，增加了成本与扩展难度。
- 技术与标准不完善：AI 场景下缺乏成熟的监控、可观测性与运维机制，行业内最佳实践尚未形成统一标准。
- 安全与合规挑战：数据隐私保护、算法偏见、模型攻击与可解释性要求日益突出。

- 成本与收益难平衡：持续监控、多模型管理和跨团队协作带来高昂成本，创新速度与风险控制需要兼顾。

为什么需要 AI 云采用框架（AI CAF）？

过去几年，阿里云通过云采用框架（CloudAdoptionFramework,CAF）帮助众多企业完成了从“上云”到“上好云，管好云”的转型。CAF 提供了一套标准化的上云、用云及管云的指导原则与最佳实践。随着大模型与 AI 应用逐步普及，企业领导层需要定义清晰明确的 AI 战略，跟企业自身的业务与组织匹配。企业的云管理团队，AI 工程团队需要搭建一套安全合规、稳定高效及易于扩展的 AI 基础设施，在面对 AI 应用高可用架构设计，成本优化，安全合规等技术挑战，也亟需有一套成熟的方法论。

为此，阿里云在通用 CAF 的基础上，推出 AI 云采用框架（AICloudAdoptionFramework,AICAF）——旨在为企业上 AI 提供策略和技术的指导原则和最佳实践，帮助企业上好 AI、用好 AI、管好 AI，并成功实现业务目标。

目标读者

本白皮书旨在为正在规划、建设或优化 AI 能力的企业，提供一套可落地、可扩展、可治理的 AI 采用方法论。通过 AICAF 框架，企业可以在 AI 工程化构建、安全合规、高可用 AI 架构、成本优化等多个维度提升 AI 应用建设的质量，真正实现从“能用 AI”走向“用好 AI”。

本书适合以下读者：

- 管理团队：包括 CEO、业务负责人，帮助他们理解企业 AI 战略，制订符合企业自身发展的 AI 路径，并搭建与之相匹配的组织团队。
- 架构团队：包括 CTO、架构师、算法研发、MLOps/DevOps 工程师，帮助他们理解如何构建高可用、可扩展的 AI 基础架构。
- 安全合规团队：包括安全审计专家、数据治理人员，帮助他们建立 AI 数据全生命周期的安全与合规体系。
- 运维团队：包括运维、监控、IT 管理人员，帮助他们利用自动化与可观测性提升 AI 系统的稳定性与运维效率。
- 业务团队：包括业务负责人、产品经理、财务人员，帮助他们在 AI 项目中平衡业务价值、成本投入与长期可持续发展。

本文件起草单位及主要起草人

指导委员会：

韩鸿源：阿里云智能集团公共云事业部首席解决方案架构师

穆 飞：阿里云智能集团研究院院长

何登成：阿里云智能集团开放平台负责人

编写组成员：（以下按姓氏拼音首字母排序）

程超、陈雪琴、曹志、盖镜源、郝伟刚、贾丁丁、赖岚、李晨、李冬萌、李玲玉、李鹏飞、林万境、娄恒、孟方、倪海峰、欧阳京、王云毅、王觯程、温曙光、吴祎凡、熊骐陟、许毅、叶剑宏、姚怡东、曾俊、张瑄、郑立异、周金龙、朱彩辉

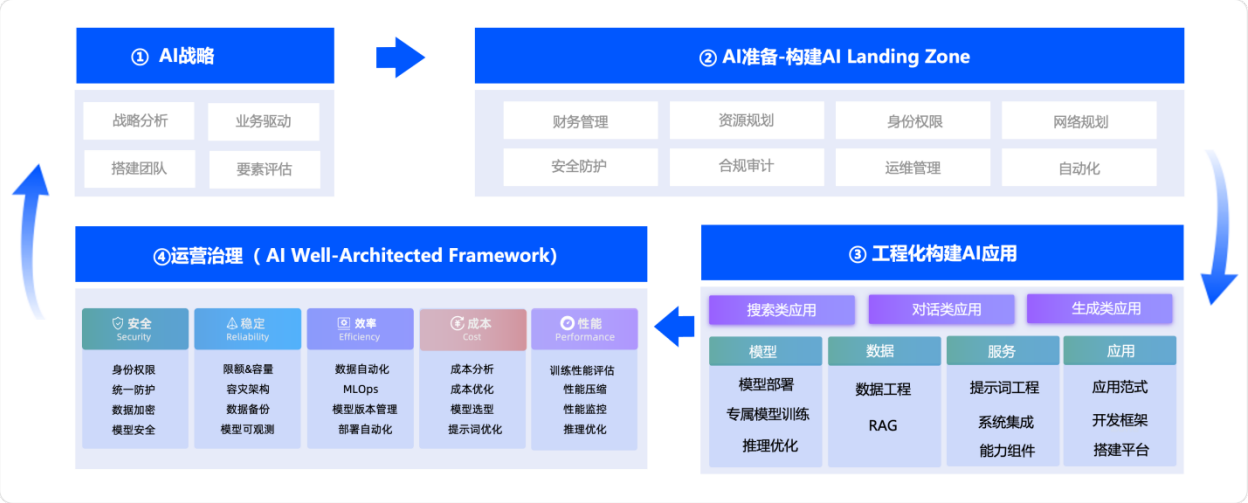
特别鸣谢：（以下按姓氏拼音首字母排序）

崔立喜、欧阳欣、任平、孙成浩、王超、王德山、魏博文、肖剑、徐栋、杨斌、张瑞、周琦、祝顺民

第二章．AI 云采用框架概述

AI 云采用框架（AI Cloud Adoption Framework, 简称 AI CAF）是一套指导组织规划、构建和运营人工智能能力的端到端方法论体系。它在通用云采用框架（CAF）的基础上，深度融合大模型应用治理与安全合规要求，帮助企业以结构化、可治理、可持续的方式实现从 AI 概念验证到规模化生产的跨越。

在云采用框架（CAF）中，将企业云采用分为四个阶段：上云战略、上云准备、应用上云和运营治理。同样企业在 AI 采用也分为四个阶段：AI 战略、AI 准备、AI 工程构建、持续运营治理。本 AI CAF 会详细探讨企业应在每个阶段采取的业务和技术策略；同时，还提供了一系列最佳实践、文档和辅助工具，帮助云管理团队、企业 AI 团队等干系人能够实现组织协同达成目标。



- 在 AI 战略阶段，领导层需要思考 AI 能为业务带来什么价值，并且寻找可以跟 AI 结合的业务场景，搭建相应 AI 团队，并在公司层面确定相应的战略。
- 在 AI 准备阶段，云管理团队需要制定在 AI 基础设施层的顶层设计，确保跟各个关联团队高效协同，并且可管可控。
- 在 AI 工程化构建阶段，围绕跟 AI 工程相关的组件进行设计，包括应该选择什么样的模型，知识库建设，提示词工程及 MCP 工具等。
- 在持续运营治理阶段，企业不断发现和解决 AI 应用在实际生产过程中的问题和风险，提升 AI 应用的安全与稳定性。

第三章．AI 战略

人工智能已经成为企业转型和创新发展的关键驱动力。在当今智能时代，企业要通过 AI 创造可持续的商业价值，就必须制定具有前瞻性的 AI 战略，以明确 AI 能力建设和技术应用方向，并通过系统化的规划和执行来推动智能化转型。AI 战略的本质是发展性的，是对企业在智能时代进行的整体性和长远性的谋划，其核心不是制定复杂的技术路线，而是回答企业在 AI 时代的生存与发展问题：包括如何对企业发展的内外形势进行科学研判，从而提出企业在智能时代的 AI 发展愿景和战略定位；探讨如何用 AI 解决业务的核心痛点、构建长期竞争力，明确不同时间段的战略目标、关键策略及路径安排，从而在市场竞争中占据主动。

AI 战略的制定，涉及业务、技术和组织的深度协作。其中，AI 战略目标必须服务于业务发展创新；AI 技术能力为战略实施提供基础底座支撑；组织保障则涵盖人才机制、跨部门协同、治理结构与文化变革，确保战略有人推、有制度护航、有文化土壤。

企业 AI 战略的整体框架如图所示：

AI 战略的制定过程包括五大关键环节：首先，确定企业 AI 发展愿景、战略定位与总体目标，为企业 AI 应用指明方向；继而进入战略整体规划和方案制定阶段，明确分阶段目标并绘制战略路线图，确定战略资源配置和投入，以及关键策略安排和实施路径；接着是 AI 应用场景筛选，聚焦高价值、可落地、有数据支撑的业务场景；随后进入落地执行和效果评估，衡量 ROI 等指标，并通过战略监控形成反馈闭环，进行迭代优化。这五个环节构成战略制定和执行的整体流程。

1. 企业 AI 愿景与战略定位

制定清晰的 AI 愿景与战略定位对企业来说至关重要，它是企业战略的核心组成部分，能帮助企业在快速变化的技术环境中保持战略定力。企业 AI 愿景和战略定位的制定依托于多维度的系统化剖析，包括对企业业务属性与现状的深入分析、对未来发展驱动力的深度思考、对行业趋势的前瞻洞察、对外部环境和竞争态势的充分研判、以及对 AI 技术潜力的全面评估等。同时，它还必须具备可操作性和普遍认同性，通过自上而下的战略共识，确保与整体业务目标保持一致，并为后续的战略实施奠定坚实基础。

1.1 内外部发展环境分析

（1）定位业务属性和驱动力

不同业务属性的企业，其 AI 的价值锚点截然不同，对 AI 的核心需求存在明显差异。例如，面向消费者市场的创新型企业，如智能汽车和消费电子公司，重点关注市场响应速度和产品创新能力，着重加强利用 AI 加速产品迭代周期；对于零售等行业，其 AI 应用的主战场是用户体验与供应链优化，通过提升客户体验和优化商品流通过程实现价值增长；对于金融业，其突出特征是风险与信任管理，通过风险定价、信用评估和合规管理创造价值；对于商业咨询服务类企业而言，AI 的主要价值在于提升内部员工效率，提升员工创造力。准确厘清企业的业务属性及发展需求，是明确 AI 愿景使命的前提。

问题清单参考

- 我们的核心业务属于哪一类型？（如：产品创新驱动型、效率领先驱动型、客户体验驱动型）
- 我们的业务是应对高频、实时的交互与变化，还是处理低频、长周期的复杂项目？AI 需要为我们提供及时响应的能力，还是深度洞察的能力？
- 我们的业务流程是高度标准化还是非标准化，需要规则明确的 AI 应用还是需要更强的情景理解和推理能力？
- 我们期待 AI 帮助我们优化现有价值链条，还是创造全新的价值来源？
- AI 技术最有可能颠覆或增强我们哪个核心业务环节？

（2）行业趋势及 AI 应用前景洞察

此外，企业也需要系统性地分析本行业与 AI 技术融合的趋势和潜力，着重关注行业技术的迭代方向、整体市场需求的变化，明确 AI 在行业各个维度的价值方向。通过分析行业数字化、智能化转型过程中的核心痛点，探讨和识别 AI 在本行业可以落地的场景边界。同时，可以研究本行业的头部企业的 AI 应用情况及商业价值实现路径，为企业战略愿景和规划提供借鉴。

（3）外部环境和竞争态势

深入分析主要竞争对手的 AI 布局策略，特别是他们在关键技术领域和应用场景方面的投入重点。通过建立竞争情报系统，企业可以及时发现市场空白点和潜在威胁。要全面研究本行业对 AI 应用的政策法规要求，及时识别潜在的风险因素，明确 AI 应用可以触达的方向，例如金融、医疗等行业受到监管的要求较高，AI 应用需优先满足可解释性、公平性和审计追踪等要求。

（4）能力成熟度评估分析

最后，企业需系统性地盘点和评估实施 AI 应用的技术能力和组织能力等相关能力成熟度，结合财务预算和规划可能的 AI 投入预算情况，从而准确定位愿景。首先需要梳理企业的数据资产整体现状，分析关键业务数据的完整性、可获取性以及治理水平，判断数据规模和质量可以支持何种水平的 AI 应用；其次需要盘点内部技术储备与人才结构，以及愿意为此所做的投入情况。这一评估过程为战略愿景目标设计提供客观依据，从而构建与企业实际能力及意愿投入相匹配的 AI 发展蓝图，避免愿景脱离实际，实施落地不及预期的情况。

问题清单参考

- 我们目前的数据基础如何？关键业务数据是否可用、可治理、可分析？
- 我们数据的特征是什么样的，以结构化数据为主还是非结构数据为主，应该选择何种 AI 技术路线？
- 我们具备怎样的技术储备与人才队伍？是否存在关键能力缺口？
- 我们为 AI 初步落地准备了多少预算？期望的投资回报周期是多久？

1.2 企业 AI 愿景确定

AI 愿景是企业发展的“北极星”，它描绘了企业期望通过人工智能技术所要达到的终极图景。这一图景需要基于深入的业务特征和内外部环境分析，锚定长期价值方向。定义企业的 AI 愿景是一个将战略思考转化为具有方向性和感召力的价值提炼过程。需要紧密结合企业的整体使命和价值观，回答希望“借助 AI，成为一个什么样的组织”。

不同性质和类型的企业，AI 愿景呈现出鲜明的差异：

垂直行业企业，AI 愿景通常更加聚焦其所在领域，旨在通过 AI 解决特定领域的根本性问题。例如，一家教育科技企业的 AI 愿景可能是：“用有温度 AI，让因材施教成为每个孩子的现实”；一家健康管理公司的 AI 愿景可能是“让 AI 成为每个人的终身健康伙伴，让主动式健康管理成为每个人的日常习惯”。

技术驱动型企业，AI 愿景往往更偏重技术突破与范式重构，或探索技术本身的最终形态，或致力于成为智能时代的基石，为个人、为社会带来的价值。例如一家专注于基础模型研究的公司，其愿景可能是“引领通向通用人工智能（AGI）的可能路径，确保其发展与人类福祉深度对齐”；一家具身智能公司，其 AI 愿景可能是“构建机器感知与行动的智能体系，让机器人走进每一个家庭与公共空间“。

综上，企业的 AI 愿景需要深刻反映其核心业务特征，并和企业整体价值观对齐，同时也能体现对本领域本行业未来发展的独特洞察。

1.3 AI 战略定位与目标

在确立了 AI 愿景后，企业需要将其转化为可细化、可执行的行动纲领。AI 战略定位旨在厘清 AI 在企业整体战略中的角色和独特价值贡献方式，回答 “AI 在企业发展中扮演什么角色、创造什么样的独特价值” 这一问题，通过聚焦资源、构建差异化能力，建立可持续的竞争优势。明确的 AI 战略定位，对企业的 AI 能力体系建设、场景选择和落地起方向指引作用，确保 AI 服务于业务核心目标。

AI 战略目标是基于 AI 战略定位，在特定时间范围内，所设定的具体、可衡量的关键成果，回答的是“在何时达成哪些具体结果”的问题。AI 战略目标需遵循“SMART 原则”即 Specific（具体）、Measurable（可衡量）、Achievable（可达成）、Relevant（与定位高度相关）和 Time-bound（时限）。企业的 AI 战略目标必须与公司的整体战略以及业务目标对齐，并且可以用量化指标衡量商业价值。

总体上看，AI 的价值贡献可以归纳为业务增收、降本增效、品牌提升和创新带动等四个方面，企业在设定 AI 战略目标时，可以围绕这四个方面进行展开：

价值维度	核心内涵	典型目标指标设置
业务增收	AI 应用直接或间接驱动的新增业务收入	AI 新产品/服务开发带来的营收增长、获取新客户数量的增长、老客户增购比例的提升、客单价提升等
降本增效	通过流程自动化与资源优化，实现的成本节约与效率提升	人力成本节约、流程处理时长缩短、错误率下降、资源利用率提升、运营成本降低等
品牌提升	对企业的整体品牌形象、客户体验和市场声誉的长期提升	客户满意度与留存率提升、市场份额增长、品牌溢价能力、客户推荐意愿等

创新带动	对企业内部创新体系、组织能力与长期竞争力的塑造	新产品上市速度、决策响应时间、知识沉淀效率、战略决策质量、员工技能提升等

围绕以上 4 大价值方向，企业应结合自身发展阶段及 AI 应用落地情况，动态设定长、中、短期的战略目标。例如在第一阶段探索期，目标应重点聚焦于基础 AI 技术能力建设与初步的降本增效验证，快速建立信心与基础；第二阶段整合期，目标设定应重点聚焦 AI 与核心业务的深度融合情况，关注其对业务增收与品牌提升的可量化贡献等；第三阶段深化期则应重点关注创新能力的持续提升和 AI 生态系统的构建，确保持续的竞争优势与行业影响力。

通过这种系统化、分阶段的战略目标设定，企业能够有效管理 AI 转型的节奏，平衡短期收益与长期战略价值，确保 AI 战略的平稳落地与持续成功。

2. AI 战略规划与关键策略

在 AI 愿景和战略目标之下，企业需要制定完善的 AI 战略规划，通过分阶段的战略任务和路径规划，逐步构建起企业的 AI 能力体系。这包括从基础的技术能力建设，到业务创新的孵化，再到组织文化的转型升级等多个维度。同时，合理的资源投入和配置策略也是确保 AI 战略成功落地的关键因素。企业需要根据自身的发展阶段和资源状况，制定科学的投资计划，并建立灵活的资源配置机制，以支持 AI 项目的持续发展和优化。

2.1 分阶段 AI 战略任务与路线图

（1）L1 探索期战略任务与路线图

在探索期，企业应着重于建立基础 AI 能力并初步验证 AI 应用的可行性和价值。这一阶段的主要目标包括：完成 AI 技术基础设施的初步搭建，培养 AI 团队和人才，以及通过小规模的试点场景项目验证 AI 应用的可行性和价值。具体而言，企业可以选择 2-3 个有明确的业务痛点、较快产生价值且可行性强的业务场景进行试点，重点关注智能客服、知识库检索等相对成熟的 AI 应用。

阶段目标	L1 探索期核心任务	
✅ 建立基础 AI 能力 ✅ 验证 AI 应用可行性 ✅ 建立信心	**场景筛选与立项**	•收集候选场景清单 •填写场景价值评分卡 •快速评估表初筛
	MVP（最小可行性产品）开发与测试	•组建项目团队（PM+技术+业务） •开发最小可行产品 •小范围用户测试 •建立基线数据测量
	试运行与推广	•正式上线并扩大用户范围 •开展用户培训 •收集反馈快速迭代
	价值验证与决策	•计算 6 个月实际 ROI •评估场景综合得分 •决定是否进入 L2 阶段

（2）L2 整合期战略任务与路线图

进入整合期，企业需要将 AI 能力深度融入核心业务流程。这一阶段的重点转向规模化部署和价值创造，主要目标包括：将 AI 应用扩展到至少 30%的核心业务流程，建立标准化的 AI 开发和运营流程。企业需要建立专门的 AI 治理机制，包括数据质量管理、算法透明度评估和伦理审查等关键环节。

阶段目标	L2 整合期核心任务	
✅ AI 深度融入核心业务流程 ✅ AI 应用规模化部署和价值创造	🔄 横向复制	将试点成功的内部提效型场景推广至其他部门
	⬇️ 纵向深化	在试点部门增加新场景，形成场景组合
	🚀 战略突破	启动产品创新型场景及能力进化型等高阶场景

（3）L3 深化期战略任务与路线图

在深化期，标志着 AI 能力已经成为企业竞争力的核心组成部分。在这一阶段，企业应致力于打造 AI 原生的组织和技术架构，让每个业务细胞具备 AI 基因。主要目标包括：实现 AI 赋能的业务模式创新，建立完整的 AI 生态系统，以及培养持续的 AI 创新能力。这一阶段的成功不仅依赖于技术能力的积累，更需要组织文化和领导力的深刻变革。

阶段目标	L3 深化期核心任务	
✅ 打造"AI 原生"的组织形态 ✅ 实现 AI 赋能的业务模式创新	🎯 战略整合	•AI 纳入年度战略规划 •调整组织 KPI 体系 •建立 AI 创新基金 •设立 AI 伦理委员会
✅ 建立完整的 AI 生态系统 ✅ 培养持续的 AI 创新能力	🚀 产品创新突破	•内部应用外部化（SaaS） •AI 能力嵌入现有产品 •全新 AI 原生产品 分阶段验证：市场验证→产品打磨→商业化→规模化

	生态体系构建	•API 市场和 MCP 建设 •构建数据联盟 •丰富应用生态
	自我演进机制	•AI 自动监控所有应用 •AI 智能预警潜在风险 •AI 资源调度动态分配

2.2 关键策略安排

围绕 AI 战略目标，企业需要制定目标推进的关键策略和路径安排，确保战略的有效落地。主要从技术能力建设、数据治理、AI 应用创新、组织升级和人才建设等方面进行安排。

（1）AI 技术能力建设

AI 技术能力建设是 AI 战略的基础支撑。企业应综合考虑业务需求、现有技术积累、预算投入、合规要求等，进行 AI 技术能力建设的路径选择。

- 对于技术能力有限的中小型企业或业务场景相对简单的组织，轻量级 AI 能力建设是最为务实的选择。这类企业可直接调用外部通用大模型的能力，通过提示词工程进行业务场景适配。
- 对于具备一定技术能力、有特定行业知识和数据积累的中型企业，定制化微调是较为理想的选择。这类企业应注重内部数据治理能力建设，提升技术人员的模型微调和 RAG 能力，同时应该具备模型应用的工程化落地能力。
- 对于大型企业、技术实力雄厚的行业领军者，自主平台建设模式则更具战略意义。这类企业可通过引进外部合作方或者自主建设 AI 技术队伍，构建统一 AI 技术平台，整合数据处理、算法开发、模型训练等核心组件，建设完整的 AI 能力中台。

（2）AI 数据治理建设

数据是 AI 应用的核心驱动力之一，其质量与规模直接决定了 AI 性能的上限。企业应该加强数据资产的建设，提升在数据存储、清洗、标注和目录化方面的能力，打通部门间的数据壁垒，为 AI 研发和应用提供安全、合规且高效的数据服务。此外，应该积极引入向量数据库、知识图谱等新型数据技术，挖掘企业内非结构化知识的价值，为检索与推理任务等提供支撑。

（3）AI 业务创新发展

可基于现有业务场景进行 AI 应用开发，也可采用内部孵化的形式进行 AI 新产品的创新孵化。整体采取“试点-推广-优化”的渐进式策略：首先，在选定业务单元开展小规模试点，通过快速原型开发和用户反馈，验证方案的可行性。试点成功之后，可将解决方案标准化、产品化，并推广至其他业务部门。在此过程中，应建立跨部门协同创新机制，例如设立创新实验室，汇聚各领域专家，共同探索 AI 应用场景与商业模式。同时，企业还需构建灵活的风险管理机制，为创新项目提供适度的容错空间与相应支持。

（4）AI 组织升级与文化

AI 创新需要适配的组织文化做支持。首先，需要对现有的组织架构进行适应性调整和创新，例如组建包括领导班子、规划部门及业务和技术关键人员在内的 AI 创新办公室，负责统筹协调全公司的 AI 相关工作。同时，在各个业务部门设置 AI 专员岗位，负责协同本部门或业务条线的创新需求，和技术部门整体对接。在文化层面，通过定期举办技术沙龙、创新竞赛、内部分享、设立专项基金等活动，激励员工积极跟踪和试用新技术，营造内部 AI 探索应用的创新氛围。

（5）AI 人才队伍建设

采取内外结合的方式进行人才引进和对外建设。内部培养方面，可以通过建立系统的培训计划，包括基础技能培训、专项技术认证和实战项目锻炼等多个层次，帮助员工逐步掌握 AI 相关技能，并加强内部经验交流。在外部引进方面，通过制定具有竞争力的人才吸引政策，吸引 AI 算法工程师、数据科学家等关键人才。此外，企业应重视既懂业务又懂技术的复合型人才培养，打破岗位之间的边界，推动 AI 应用项目的顺利推进。

2.3 资源投入与配置规划

资源投入与配置规划是确保 AI 战略成功落地的重要保障，合理的资源分配和优先级设置，可以更高效地推动 AI 战略实施。

自建与外部供应商引入：企业需要明确哪些需要内部自建，哪些需要引进外部供应商和合作伙伴。内部自建的主要优势在于对核心技术的完全掌控，从而减少对外部供应商的依赖，适用于希望将 AI 作为核心竞争力打造且 AI 应用场景复杂的企业。而外部合作则能够帮助企业快速获取先进的 AI 技术产品和解决方案，从而缩短产品/服务上线和应用部署时间。

资源配比与优先级设定：为了确保资源投入的高效性，企业需要按照业务优先级排序制定资源配比方案。如果企业的目标是通过 AI 提升市场竞争力，则应优先投入在 AI 推动技术创新和产品研发的领域；如果目标是迅速实现降本增效，则资金应优先投入在流程优化等领域。企业可以创新资源配置机制，比如建立内部“AI 预算池”，将 AI 预算分为基础预算和弹性预算两部分。其中，基础预算用于支持核心技术、基础设施及人才的持续投入，确保 AI 能力的基本保障；弹性预算部分可以根据市场需求变化和项目进展情况进行动态调整；或者可以为有亮点突破的创新项目提供及时快速的支持。

综上，企业在 AI 战略的资源规划中，需要从愿景出发，结合自身的业务目标和外部市场环境，制定科学合理的资源配置方案，并通过持续的评估和调整，确保资源投入的有效性。

3.AI 应用场景选择与评估

AI 应用场景选择与评估是企业 AI 战略的核心环节，是 AI 技术能力与业务需求精准匹配的过程。不仅需要充分评估技术可行性，更要深入理解业务痛点、价值潜力以及与企业战略目标的契合度。

3.1 AI 应用场景分类

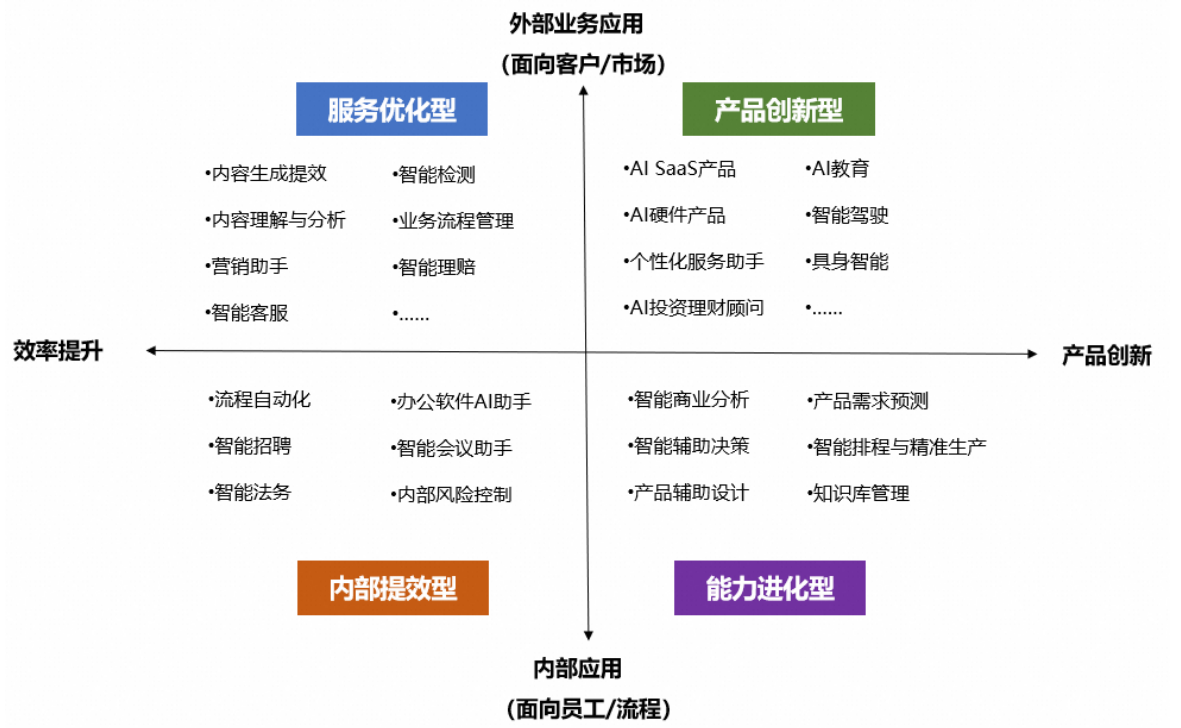
AI 应用场景类型多样，从面向对象看，既包含面向内部员工和流程的相关应用，也有面向外部客户和业务市场的场景应用；从 AI 产生的价值来看，既包含通过生产力工具实现效率提升，也包含通过创造新产品、新模式实现经营增长。因此，我们根据应用面向的对象和产生的价值路径两个维度进行划分，构建 AI 应用四象限矩阵：横向维度包括效率提升与产品创新两个方向，代表 AI 应用的不同价值路径；纵向维度包括面向员工/流程的内部 AI 应用以及面向客户/市场的外部业务应用，代表 AI 应用的不同应用对象。横纵结合形成四个象限，可以总结为：内部提效型、能力进化型、服务优化型和产品创新型。

内部提效型 AI 应用，更加侧重企业的内部工作流程优化，提升员工生产力与组织敏捷性，实现降本增效；这类应用不太可能为企业带来市场差异化，但对于以人力资本为主要竞争力的商业服务类企业来说提效显著。

能力进化型 AI 应用，更加侧重通过 AI 优化内部知识体系沉淀，提升内部创新能力，重构内部决策机制，构建智能化的组织底座，这类应用对于研发创新型及具备复杂业务系统的机构/企业至关重要。

产品创新型 AI 应用，更加侧重在通过 AI 驱动新产品、新服务和新商业模式，构建差异化的企业市场竞争力，提升创造力，这对面向消费者侧的企业意义重大。

服务优化型 AI 场景应用，则更加侧重于提升面向客户或合作伙伴的运营流程效率，优化客户服务体验，降低交付成本。



企业可以结合业务属性及战略目标进行潜在场景选择、应用场景组合构建及优先级排序。例如，成本敏感的企业可优先聚焦内部提效与服务优化；产品驱动的企业将产品创新作为核心，并以能力进化作为支撑；客户服务为主的企业则需重点强调服务优化，并辅以内部提效来释放前端资源。

3.2 基于“价值-可行性”的潜在应用场景选择

AI 应用的潜在场景可以基于“价值-可行性”进行挖掘识别，通过业务需求分析、痛点问题梳理，找到 AI 应用的关键切入点及可能产生的价值；并通过技术实现、数据获取的难易程度评估场景实现的可行性。企业可以通过以下标准化的快速评估流程对潜在场景进行初步筛选，形成决策漏斗。

筛选维度	具体标准	通过条件
痛点清晰度	能否用一句话清晰描述要解决的问题？ 该问题是否高频发生？	✓ 是
数据可用性	是否有 6 个月以上的历史数据？ 数据质量是否达到可用标准？	✓ 是
实施可行性	现有系统是否支持 API 对接？	✓ 是/可改造

	实施周期是否≤6 个月？	
价值可量化性	能否设定 3 个以上量化指标？ 预期年节约≥50 万或增收≥100 万？	✓ 是
风险可控性	失败后是否影响核心业务？ 是否有应急备选方案？	✓ 否

通过这套评估机制的场景，可以作为潜力候选对象，进入下一阶段的选择矩阵和优先级排序。

3.3 AI 应用场景优先级排序

筛选出潜在应用场景后，建议采用业务价值-实施可行性的矩阵作为分析工具进行优先级排序以指导资源分配。以“业务价值”为纵轴，以“实施可行性”（综合技术、数据和组织因素）为横轴，交叉形成 4 个象限，每个象限对应不同的战略价值和投入策略。

对于“战略重点”象限的场景，企业应集中优势资源投入，作为首期核心的 AI 应用场景打造企业标杆项目，验证 AI 价值；对于“储备攻坚”象限的场景，企业可以中长期分步骤投入，需要基于愿景目标保持战略定力和耐心；对于“轻量试行”象限的场景，因为业务价值较低属于外围项目，企业可以作为培养团队、验证技术平台的练手项目，但应避免过度占用核心资源。而对于“谨慎规避”象限的场景，除非有极强的战略或合规理由，否则应果断舍弃，避免资源浪费。

	高实施可行性	**低实施可行性**
高业务价值	**象限一：战略重点** **策略：**立即启动，优先分配资源 **特征：**价值明确，技术成熟，能快速验证战略方向 **案例：**在线教育中基于大模型的个性化伴学与教学过程增强，在为学生提供实时、个性化学习反馈的同时，显著减轻教师工作量，提升教学质量和机构运营效率	**象限二：储备攻坚** **策略：**规划立项，分阶段投入，攻克瓶颈 **特征：**价值较大但技术或整合难度高，需长期投入 **案例：**金融业中的智能投顾系统，需突破数据整合、合规审批与业务模式变革等多重障碍

低业务价值	**象限三：轻量试行** **策略：**敏捷试行，积累经验，验证技术 **特征：**易实现、见效快，是能力建设的“练手”场景 **案例：**企业内部智能招聘助手，辅助 HR 和业务部门进行候选人简历初筛、自动生成个性化面试问题建议并对候选人面试表现进行总结提炼	**象限四：谨慎规避** **策略：**除非有极强的战略或合规理由，否则明确拒绝，不予投入 **特征：**价值与可行性双低，投资回报率不明确 **案例：**为边缘业务线开发复杂的 AI 预测模型，既无法创造显著收益，又需要大量数据工程工作

3.4 构建“事前-事中-事后”的应用价值评估机制

AI 应用价值评估应该贯穿事前筛选评估、事中追踪评估和事后复盘评估三个阶段，并采用相对统一的指标框架和体系，但在评估重点、数据来源和决策用途上可以有所差异，不同阶段的评估目标、评估工具及决策输出等如下表所示。

评估阶段	🎯 核心目标	数据来源	🛠️ 评估工具	⏱️ 评估频率	✅ 决策输出
事前筛选评估	•识别高潜力场景 •预测预期价值 •形成投入分配依据 •降低决策风险	•行业基准数据 •内部历史数据 •专家经验判断 •竞品案例分析	•场景评估表 •价值预测分析 •四象限工具	**一次性评估** 项目启动前	**≥70 分** **项目启动** **60-70 分** **优化方案** **<60 分** **暂缓/放弃**
事中追踪评估	•监控价值发挥 •及时发现偏差 •动态调整策略 •控制项目风险	•系统运行数据 •业务实际表现 •用户反馈信息 •阶段性收益	•指标仪表盘 •三色预警机制 •偏差分析工具	**月度/季度** 实施过程中 持续监控	🟢 **绿灯** **继续推进** 🟡 **黄灯** **改进计划** 🔴 **红灯** **熔断机制**
事后复盘评估	•衡量整体项目情况 •沉淀可复用知识 •优化评估模型 •指导后续项目	•完整周期数据 •全面业务表现 •深度用户访谈 •财务结算数据	•价值实现报告 •最佳实践文档 •案例库系统	一次性评估 项目完成后 或里程碑节点	**•技术组件复用清单** **•流程模板标准化** **•知识库更新** **•案例分享文档**

事前评估可以帮助企业预测预期价值，识别高潜力的场景，从而做出正确的投资决策，避免在低价值场景上浪费资源。可通过填写场景价值评分卡，对可选应用场景进行预测性评分。这个阶段虽然依赖预测性数据或行业对比数据，准确性相对较低，但仍然能够为场景筛选和决策提供辅助支持。

事中评估通过实时监测项目进展，帮助企业能够及时纠偏，提高项目成功率。在这一阶段，企业需要建立月度/季度的评估机制，定期更新关键指标的动态值。通过对比动态的实际得分与预期得分，可以发现一些表现良好或者超预期的亮点项目及低于预期的场景，从而及时调整采取相关应对策略。

当一个场景完成实施周期或达到某些重要的里程碑节点时，需要进行全面的事后评估，盘点整体的成本收益，并基于评估结果进行模型的校正，以提升下一轮项目的预测准确性和实施效率。更重要的是，事后评估能够将项目实施落地过程中的成功经验和失败教训沉淀为可复用的知识资产。

3.5 AI 场景应用价值评估体系

在构建企业 AI 应用的评估体系时，不同应用场景的核心诉求和价值导向存在显著差异，需通过差异化指标体系进行精准衡量。基于业务目标与价值创造路径的多样性，我们对四大场景类型分别给出不同的衡量指标体系，但下列表格不能穷尽各类 AI 场景的评估指标，可作为同类跨行业场景的结构化、可量化的参考框架，其他场景指标可进行参考并做相应调整。

（1）内部提效型评估指标

核心原则：指标设计关注降本增效效应，聚焦效率提升、成本节约与员工体验改善等。

评估维度	关键指标	如何测量
效率提升	•流程自动化率 •任务处理时间减少率 •人力成本减少率	•自动化任务数/总任务数×100% •(原时间-新时间)/原时间×100% •(原人力成本-新人力成本)/原人力成本×100%
成本效益	•ROI 计算 •投资回收周期	•(总节约成本-投入成本)/投入成本×100% •成本回收所需的时间（月）
质量保障	•错误率变化 •异常处理能力 •风险识别率提升	•(原错误率-新错误率)/原错误率×100% •异常状况自动解决率 •发生风险的损失成本减少
员工体验	•员工采纳比例 •智能化内容生成采纳比例 •工作满意度变化	•使用人数/目标人数×100% •智能化内容生成采纳比例（如文本、代码和图像生成占比） •满意度调查对比

（2）能力进化型评估指标

核心原则：指标设计关注创新驱动效应，聚焦决策质量、内部创新能力和组织协同能力。

评估维度	关键指标	如何测量
创新能力	•创新提案增长数量 •知识产权取得提升 •战略目标达成率提升	•月均提案数量增长 •专利及论文发表增长提升比例 •(新达成率-原达成率)×100%
知识沉淀	•知识资产增长率 •最佳实践复用率 •组织学习效率	•(新知识量-原知识量)/原知识量×100% •复用案例数/总案例数×100% •培训周期缩短率
决策能力	•决策速度提升 •高管使用频率 •数据驱动决策比例	•(原时间-新时间)/原时间×100% •高管周使用次数 •数据决策数/总决策数×100%
组织适应	•跨部门协作效率 •数据共享程度 •AI 技能人才占比	•项目协作满意度评分 •共享数据源数量 •AI 人才/总人数×100%

（3）服务优化型评估指标

核心原则：指标设计关注客户体验、运营效率提升及带来的业务影响。

评估维度	关键指标	如何测量
客户体验	•客户满意度变化 •问题解决率 •交互自然度	•净推荐值（NPS）或客户满意度（CSAT）对比 •自动解决问题/总问题×100% •用户体验评分（1-5 分）
运营效率	•服务成本降低 •响应速度提升 •人力需求减少	•(原成本-新成本)/原成本×100% •(原时间-新时间)/原时间×100% •(原人力-新人力)/原人力×100%
业务影响	•客户留存率提升 •交叉销售率提升 •新客户获取成本变化	•(新留存率-原留存率)×100% •交叉销售订单/总订单×100% •(新成本-原成本)/原成本×100%
系统可靠性	•系统可用性 •故障恢复时间 •合规性保障	•正常运行时间/总时间×100% •平均恢复时间（分钟） •合规检查通过率

（4）产品创新型评估指标

核心原则：指标设计偏重创新驱动和业务增长效应，关注市场价值与创新突破。

评估维度	关键指标	如何测量
商业价值	•新业务收入占比 •客单价变化 •ROI	•AI 产品收入/总收入×100% •(新客单价-原客单价)/原客单价×100% •(收入-成本)/成本×100%
产品竞争力	•功能独特性评分 •同类产品市场份额对比 •用户粘性	•专家评分（1-5 分） •市占率提升 •月活跃率、留存率
市场接受度	•用户增长率 •付费转化率 •NPS 净推荐值	•月均用户增长百分比 •付费用户/总用户×100% •(推荐者-贬损者)/总样本×100%
创新可持续性	•新功能上线速度 •数据飞轮效应 •生态系统建设	•月均新功能数量 •数据量月增长率 •合作伙伴数量

4. AI 战略监控与迭代

AI 战略应该采用动态闭环的管理机制，通过持续的数据监控收集、分析和反馈，确保战略能够灵活适应不断变化的技术环境和业务需求。

4.1 AI 战略动态监控

战略监控体系是组织的战略仪表盘，通过对海量运营数据的观察监控，识别出可能影响战略健康度的关键信号，分析 AI 战略推进态势、价值实现路径及存在的潜在风险。通常需要监控业务价值、技术效能、组织能力与创新以及风险与合规等 4 个方面。

业务价值层面，监控的重点是 AI 投入的最终价值产出，可以和上述 AI 场景应用的事中事后评估结合起来，着重衡量 AI 应用带来的财务价值和战略价值，如 AI 驱动产品/服务带来的营收增长、通过 AI 优化实现的运营成本节约以及 AI 应用带来的客户满意度或市场份额提升等，这些指标是体现 AI 价值的最直观的数据。

技术效能层面，监控的重点是 AI 应用质量、效率与稳健性，包括 AI 应用在生产环境中的性能指标（如准确率等）、系统服务的可靠性（如 API 可用率、平均无故障时间）以及资源利用效率（如模型推理成本、计算资源利用率等）。技术效能的健康度直接决定了 AI 能否规模化、可持续地交付商业价值。

组织能力与创新层面，监控重点应涵盖人才梯队建设（如 AI 人才保有量、技能提升进度）、AI 文化渗透度（如各部门自主发起 AI 的项目情况，AI 相关学习活动组织情况）以及创新活力（如年度概念验证项目数量、项目成功率）。组织层面的指标是先导性的，能够为未来 AI 的持续创新带来潜能。

风险与合规层面，企业需要监控模型公平性与偏差指标、数据隐私与安全合规状态、以及模型鲁棒性与对抗性攻击的能力等，确保 AI 应用及产出符合法律法规、行业规范及道德规范，避免潜在的偏见风险和不良社会影响。

战略监控过程中应重点防范以下四类风险：

风险类型	主要表现	预防措施	应对策略
⚠️ **价值实现不及预期**	•ROI 持续为负 •关键指标改善不明显 •用户采纳率低 •业务价值难以量化	✓事前充分评估 ✓分阶段投入 ✓快速迭代机制	制定改进计划 及时止损转向高价值场景
🔒 **数据质量与安全**	•数据质量不高 •敏感数据泄露 •数据合规问题 •AI 决策错误	✓数据质量评分 ✓数据分类分级管理 ✓定期安全审计 ✓建立合规检查点	技术措施：数据脱敏加密 制度措施：访问权限体系 应急措施：泄露应急预案
👥 **组织变革抵触**	•部门利益冲突 •缺乏 AI 技能 •推进困难阻力大	✓建立激励机制 ✓全员 AI 素养培训 ✓提供转岗支持	优先开放部门试点 树立成功标杆
⚖️ **合规与伦理**	•AI 决策存在偏见 •侵犯用户隐私 •违反监管要求 •缺乏透明度	✓建立伦理审查委员会 ✓高风险应用人工复核 ✓跟踪监管动态 ✓提前布局合规	制定 AI 伦理准则 建立可解释 AI 机制 定期合规审计

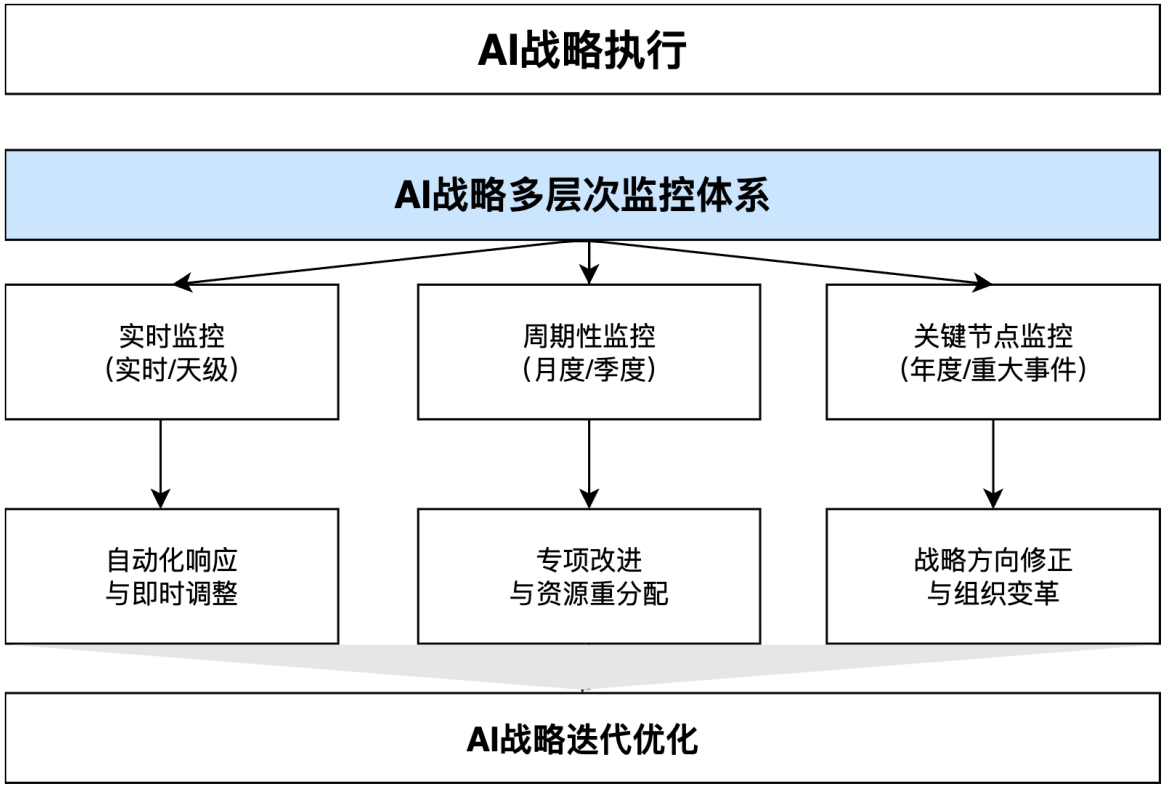
4.2 反馈与调整机制

通过实时和定期监控上述四类战略指标后，企业需要建立反馈和调整机制，推动战略动态优化，确保 AI 战略始终保持活力和竞争力。

实时监控和问题追踪：企业需要构建自动化告警系统，设置关键监测指标的阈值，在异常情况出现时能够及时响应。例如，电商平台通过部署智能监控系统，能够在 AI 推荐引擎的点击率出现异常波动时，第一时间触发告警并启动问题排查流程。

周期性调整和专项改进：企业需要对 AI 应用项目进行周期性的评估，分析项目的进展情况、资源使用效率以及投资回报率。例如，某汽车制造商在每季度的战略审查中，都会对 AI 驱动的自动驾驶辅助系统的开发进度进行详细评估，并根据评估结果调整资源分配和执行计划。

定期审视及战略迭代：企业需要在年度或者关键节点重新评估战略方向、调整长期发展目标。例如，某医疗健康公司在年度战略审查中，决定将 AI 技术的应用重点从运营效率提升转向产品创新，推出了基于 AI 的个性化诊疗方案。此外，如果外部环境发生重大变化时，例如新技术的突破或市场需求转型等，企业需要快速响应及时调整战略，以适应新的竞争格局。



5.小结

企业 AI 战略的本质是面向智能时代，通过系统性 AI 发展规划，推动技术、业务与组织的协同创新，实现可持续的价值创造。AI 战略制定需要围绕五大关键环节展开：从愿景与目标出发，经整体规划、场景筛选、落地执行到效果评估、战略监控迭代，形成闭环的动态流程。其中，业务导向是核心驱动力，技术能力是底座基础，组织保障是落地支撑。

第一，明确愿景与目标，这是战略制定的起点。企业需精准识别内外部环境变化，结合自身业务属性和发展需求，明确 AI 的价值锚点。通过构建适配自身战略的 AI 愿景和目标，确保 AI 应用方向与长期价值创造路径高度对齐。

第二，开展整体规划与关键策略安排，这是战略落地的关键。企业需在总体战略定位和目标下，系统设计分阶段目标、核心任务和路线图，明确技术选型、数据治理、团队搭建等关键策略，通过科学的资源配置计划，将战略目标分解为可执行的里程碑任务。

第三，聚焦场景筛选与组合，这是价值实现的核心抓手。企业需基于业务痛点和数据基础，筛选高价值、可落地的 AI 应用场景，平衡短期收益与长期潜力。通过构建场景优先级矩阵，优先部署能快速验证价值、带动全局变革的标杆案例，并逐步复制推广，纵深推进。

第四，推动落地执行与效果评估，这是战略闭环的保障。企业需建立跨部门协作机制，确保技术、业务与组织的高效协同。通过构建“事前-事中-事后”的应用价值评估机制，定期监测 AI 应用成效，并基于效果评估，动态调整资源配置和实施路径。

第五，构建战略监控与迭代机制，这是战略演进的必然要求。实时和定期监控各类战略指标，复盘内外部环境变化，建立反馈和调整机制，推动战略动态优化。

综上，AI 战略的制定与执行需要企业以系统性思维统筹全局，在动态平衡中实现技术赋能与业务创新的深度融合，打造在智能时代的竞争优势。

第四章．AI LANDING ZONE

1. 资源规划

1.1. 概述

资源规划是构建高效、经济且可扩展的 AI Landing Zone 的起点。一个深思熟虑的规划不仅能为复杂的 AI 工作负载提供稳定支撑，更是实现安全隔离、成本归因和敏捷治理的基石。本章旨在为您提供一套系统性的资源规划框架，涵盖多账号体系、MaaS/PaaS/IaaS 三层平台资源配置，以及通过资源组与标签实现精细化治理的最佳实践，确保您的 AI 基础设施从一开始就具备坚实的基础。

1.2. 背景与挑战

AI 业务的资源规划，需要在多变的业务需求、高昂的资源成本和严格的安全合规之间寻求平衡，这带来了四大核心挑战：

- **安全隔离 vs. 协同效率**：在多团队、多项目并行的 AI 环境中，如何既能通过账号、工作空间等机制实现严格的资源和数据隔离，防止互相干扰，又能保障跨团队之间安全、高效的数据与模型共享，是组织层面的首要挑战。
- **资源争抢 vs. 成本效益**：以 GPU 为代表的 AI 算力资源既稀缺又昂贵。如何在保障核心业务（如在线推理）性能的同时，让多个训练任务公平、高效地共享算力池，并最大化资源利用率，避免资源闲置或争抢，是资源调度的核心难题。
- **架构多样性 vs. 管理一致性**：企业可能同时使用 MaaS、PaaS、IaaS 等多种模式构建 AI 应用，每种模式都有其独特的资源组织和管理范式（如百炼的工作空间、PAI 的资源配额、ACK 的集群与节点池）。如何在此之上建立一套统一的、贯穿所有平台的治理和监控策略，避免管理碎片化，是一大挑战。
- **成本归属模糊 vs. 精细化运营**：AI 任务会消耗计算、存储、网络等多种资源，若缺乏有效的组织和标记，海量的云上成本将难以精确分摊到具体的业务线、项目或团队。这种“糊涂账”使得成本优化和 ROI 评估无从下手，严重阻碍了 AI 业务的精细化运营。

1.3. 具体方案

1.3.1. 多账号规划

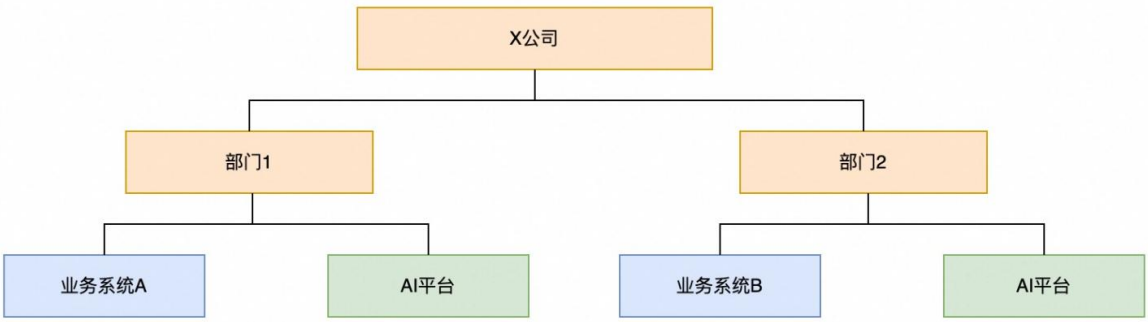
企业在构建人工智能能力的过程中，合理的云上组织与账号结构设计是实现安全隔离、成本治理、权限控制和合规审计的基础。基于阿里云资源目录（ResourceDirectory）和多账号架构（Multi-

AccountArchitecture），本节提供一套面向 AI 场景的标准化多账号设计原则与示例。以下是多账号设计参考原则：

- 区分生产、非生产环境。建议将不同环境部署在不同账号内。
- 不同部门或者不同项目如果需要在权限与资源做强隔离，建议部署在不同账号内。
- 不在管理账号内部署业务用到的云资源。管理账号只用作管理，避免权限过大导致的越权风险。
- 管理账号内要做好身份安全设计，开启 MFA 认证，避免出现管理员身份泄露。
- 多账号使用统一登录，提升人员登录账号效率。
- 随着业务发展，需要将业务部署在新账号。推荐使用账号工厂，快速创建安全合规的新账号。

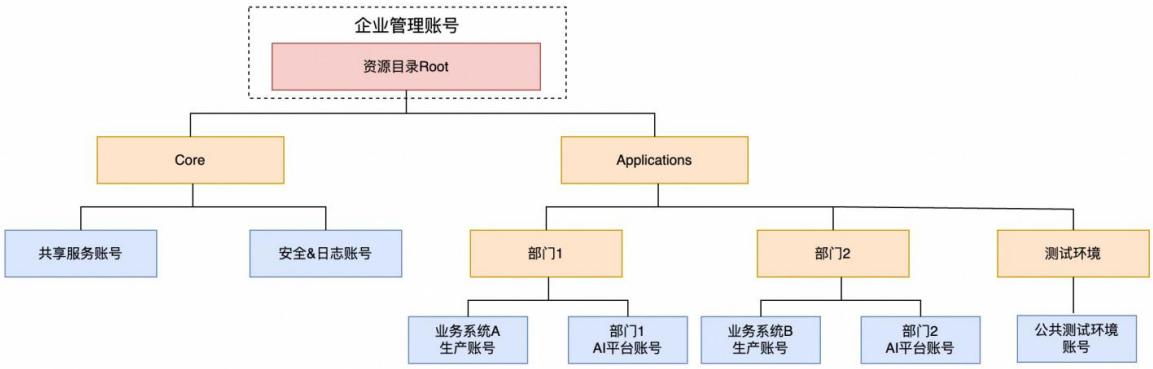
(1) 示例

X 公司有多个业务系统，由不同的团队管理。公司最近在规划 AI 战略，不同团队都有各自的 AI 项目准备部署到阿里云。以不同云账号为单元来承接不同团队业务与 AI 系统。各个业务团队根据自己职能团队角色获得相应权限。



接下来，我们将为您介绍多账号资源管理体系建立的设计思路与建议。

(2) 多账号结构规划



(3) 设计建议

- 参考标准 Landing Zone 多账号实践，设计原则参考：

- 非业务相关的资源放在 Core 资源夹下，包括安全、日志、共享服务等账号。
- 业务相关的资源，建议按不同部门进行划分。其中测试环境可以放在一个大的测试账号内，做到资源利用率最优。生产环境建议可以按不同业务系统进行隔离。AI 平台相关资源建议部署在一个单独账号，以实现资源与人员权限隔离。
- 企业管理账号为资源目录的超级管理员，建议不要将其用于资源目录管理之外的其他任何用途。妥善管理此账号，并设置 MFA 双重验证，加强安全访问管理措施。

1.3.2. AI 平台资源规划

(1) MaaS（百炼）工作空间规划

在使用阿里云百炼平台时，“业务空间规划”是指对企业内部不同团队、项目、环境、数据和模型的逻辑与物理隔离机制的设计，通过工作空间隔离可以达成如下目标：

- 实现多部门/团队协同而不互相干扰
- 满足安全隔离与权限控制要求
- 支持从 POC 到生产的全生命周期管理
- 便于成本分摊与治理审计

工作空间规划不仅是技术问题，更是组织架构与流程治理的体现。建议从以下两个维度进行系统性规划：

- 组织级空间划分（Organization&Workspace）
- 环境隔离设计（Dev/Staging/Prod）

维度一：组织级空间划分

常见 Workspace 划分推荐：

- 按组织部门划分，适用于跨部门协作，责任明确。比如人力资源部 AI 助手，财务部门智能报表系统，客服部门问答机器人，就可以定义三个业务空间，分别分配给人力资源部、财务部、客户部。
- 按业务线划分，适合多产品线独立运营场景。比如某银行企业有信用卡业务、助贷业务、个人理财业务等，就可以按不同业务线分多个业务空间。
- 按项目来划分，适合于临时验证项目，比如需要做个技术 POC，那可以单独划一个 POC 的工作空间出来。

最佳实践建议：

- 每个业务团队应拥有独立的 Workspace，避免配置冲突和权限越权
- 禁止“All-in-One”大 workspace，防止治理混乱

维度二：环境隔离设计

为保障 AI 应用稳定上线，必须建立标准化的环境隔离机制。推荐架构：

- 开发环境（Dev），适合于功能开发、Prompt 调优、知识库测试。这个环境的特点包括：允许频繁变更、可使用测试模型、不会对接生产数据。
- 预发环境（Staging），适合于集成测试、性能压测及 UAT 验收。
- 生产环境（Prod），对外提供正式服务，这个环境特点：严格变更审批，启动私有网络访问（PrivateLink）、实时监控 SLA（延时、错误率等）

1.3.3. PaaS（PAI）工作空间与算力规划

(1) 工作空间规划

PAI 工作空间是一个资源与权限的逻辑隔离单元，用于组织和管理 AI 项目的全部资产。建议按组织级划分不同工作空间，常见 Workspace 划分推荐：

- 按组织部门划分，适合于跨部门独立运营。比如某知名保险企业，有两个相对独立的保险子部门，相关算法工程师及 AI 算力资源都是独立的。这种场景就适合按照部门来划分业务空间。
- 按业务线划分，适合于多产品线并行。如某游戏企业有多款游戏，不同游戏的算力成本是独立核算的。这种场景就适合按不同产品线来划分业务空间。
- 按项目划分，适合于临时验证项目，比如需要做个技术 POC，那可以单独划一个 POC 的工作空间出来。

最佳实践建议：

- 禁止“All-in-One”大 workspace，防止权限泛滥和成本混淆。

(2) 训推一体算力规划

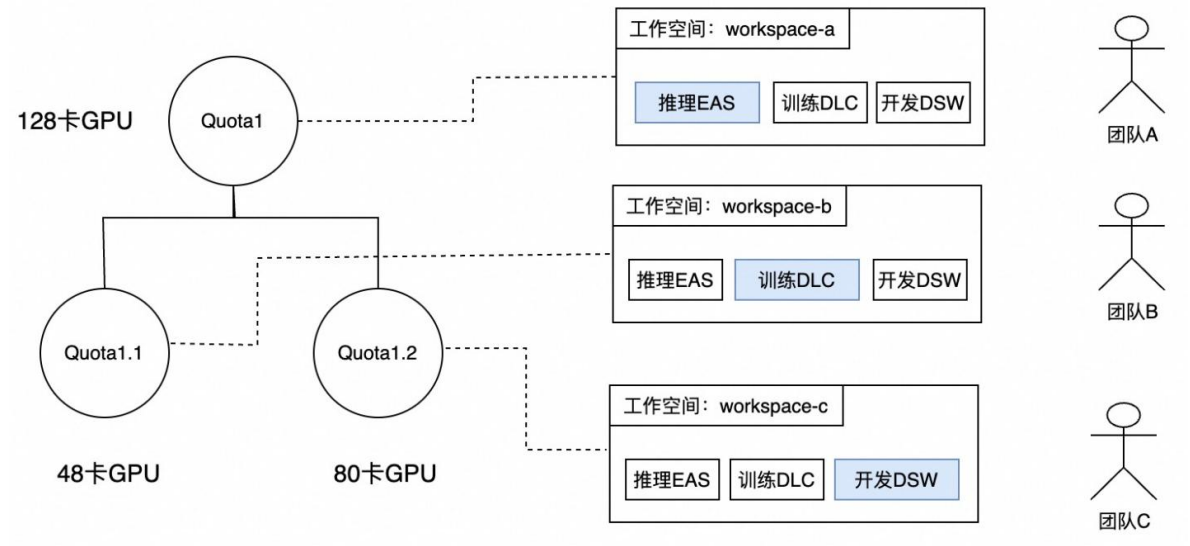
在多团队协作环境中，合理分配计算资源是确保各团队高效运作的关键。

假设购买 AI 计算资源（例如总计 128 卡 GPU），用于 A、B 和 C 三个团队。其中：

- A 团队负责推理服务，需要高资源保障。
- B 团队和 C 团队分别是训练团队，用来提交训练任务。
- B 和 C 团队的训练任务相较于 A 团队的推理服务优先级更低。即当 A 团队推理资源不足时，系统可以快速回收用于训练的资源，优先满足推理服务的需求。
- B 和 C 团队使用的计算资源量可调整，可以根据实际需求动态增加或减少资源。
- B 和 C 团队可以管理各自的资源和任务。

(3) 方案介绍

参考工作空间规划，按不同组织进行工作空间划分。



方案如下：

- 创建资源配额 Quota1（例如 128 卡 GPU），并打开子级算力抢占开关。然后为 Quota1 创建两个子级资源配额，分别为 Quota1.1（例如 48 卡 GPU）和 Quota1.2（例如 80 卡 GPU）。如上图所示，Quota1 与 Quota1.1 和 Quota1.2 形成父子级关系 QuotaTree，其中 Quota1 为父级资源配额，Quota1.1 和 Quota1.2 为子级资源配额。
- 为团队 A 创建工作空间 workspace-a，并绑定 Quota1。在 Quota1 上部署 EAS 服务，用于模型推理。
- 为团队 B 创建工作空间 workspace-b，并绑定 Quota1.1。在 Quota1.1 上创建 DLC 任务。
- 为团队 C 创建工作空间 workspace-c，并绑定 Quota1.2。在 Quota1.2 上创建 DSW 实例，进行模型开发。

1.3.4. IaaS（ACK）集群规划

在企业级 Kubernetes 环境中，ACKPro 集群是阿里云为生产环境设计的高可靠、强安全、可治理的托管 Kubernetes 服务。其规划不仅涉及底层基础设施，还包括上层资源组织（如 Namespace）的设计。

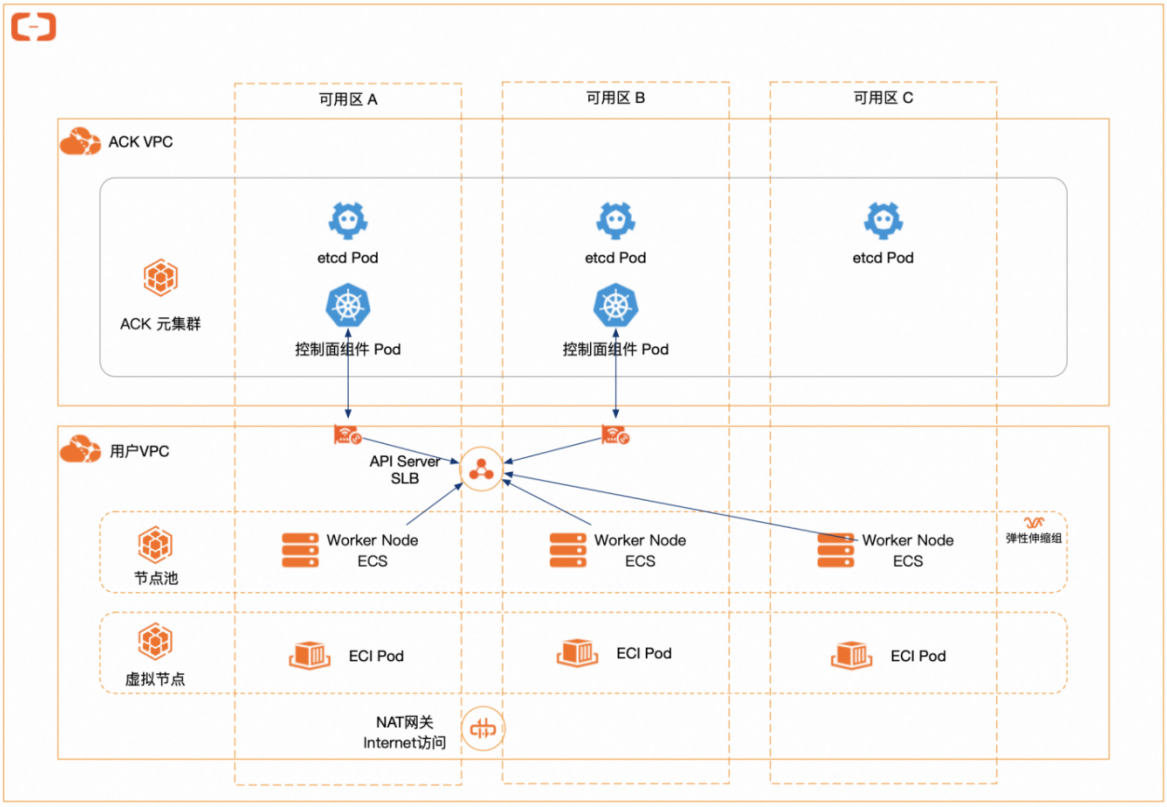
(1) 集群规划参考原则

原则 1：按业务域或环境隔离集群

一个 ACKPro 集群应服务于单一业务域或环境，避免混合用途。切忌所有微服务共用一个大集群，有可能会带来“雪崩风险”与“权限管理混乱”。

原则 2：启用多可用区（Multi-AZ）部署控制面与工作节点

ACKPro 默认支持跨 3 个可用区部署 Master 节点，确保控制平面 99.95%SLA。工作节点 NodePool 按 AZ 划分实现节点故障自动迁移，流量就近接入。



原则 3：合理规划节点池（NodePool）与弹性策略

使用多节点池加弹性策略，以实现资源高效利用。

(2) Kubernetes Namespace 规划参考原则

Namespace 是 Kubernetes 中逻辑隔离的基本单位，需科学设计以支持多团队协作与治理。

原则 1：按环境+业务线二维矩阵划分

参考示例

Namespace	说明

prod-llm-serving	生产环境大模型服务
staging-recommendation	预发推荐系统
shared-monitoring	公共监控组件（Prometheus/SLS）

不建议使用模糊的名称如 default，test 来命名，也不建议用一个 Namespace 来承载多个项目。

原则 2：每个 Namespace 配置独立配额（ResourceQuota）与限制范围（LimitRange）

通过给每个 NS 设置独立配额，可以防止资源滥用影响到其他 NS。例如，默认情况下，运行中的 Pod 可以无限制地使用节点上的 CPU 和内存资源，这意味着某个命名空间的 Pod 可能会耗尽集群的资源。此时，您可以为命名空间配置资源配额额度，包括 CPU、内存、Pod 数量等。

原则 3：命名空间与网络、存储策略联动

功能	实现方式
网络隔离	使用 terwayNetworkPolicy 做东西向 pod 网络隔离
存储隔离	PVC 绑定到特定 StorageClass（如极速型 NAS）
日志归集	SLS 采集
监控告警	ARMSPrometheus 按 NS 聚合指标，设置独立告警规则

1.4. 采用资源组与标签来管理 AI 服务依赖的资源

在企业构建和运营人工智能（AI）系统的全生命周期中，随着模型训练、推理部署、数据处理等任务的不断扩展，AI 服务所依赖的云资源数量呈指数级增长。这些资源分散于计算（ECS/GPU）、存储（OSS/NAS）、网络（VPC/SLB）、平台服务（PAI/百炼）等多个维度，若缺乏统一的组织与治理机制，极易导致：

- 资源归属不清，成本无法分摊
- 权限混乱，安全合规风险上升

- 自动化流程难以实施

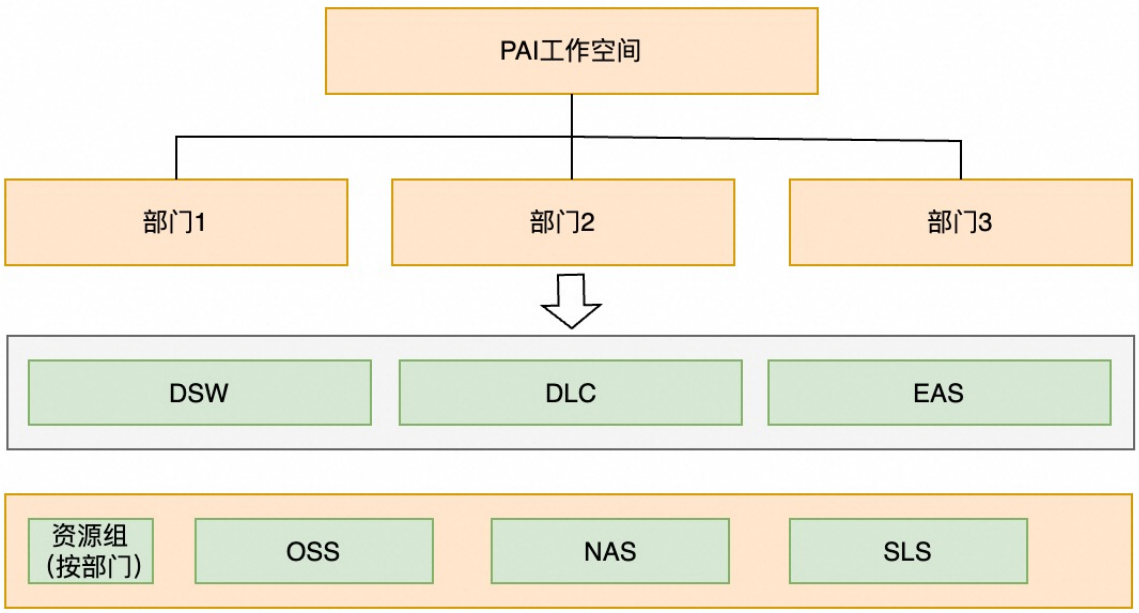
为此，资源组（ResourceGroup）与标签（Tag）成为阿里云上实现精细化资源管理的核心手段。

1.4.1. 资源组：AI 资源的一级组织单元

资源组（ResourceGroup）是阿里云提供的一种资源组织机制，用于将同一类资源归集到一个逻辑容器中。建议每个资源组对应一个业务团队或者项目，并绑定一个成本单元。

(1) 示例

X 公司使用阿里云 PAI 来做训推平台，按照不同部门来设计 PAI 里面的工作空间，每个部门用到的 AI 依赖资源都是需要严格控制成本与权限。参考设计方案：



方案如下：

- 按照部门来规划 PAI 里面的工作空间，通过工作空间隔离资源权限。
- 规划不同部门的资源组，用于将 PAI 平台关联的资源划到指定的资源组。
- 在阿里云费用中心设置分账规则，按资源组定义分账规则。
- 配合 RAM 授权策略，限制某 RAM 用户只能查看/操作指定资源组的资源。

特殊限制

阿里云支持资源组的产品

1.4.2. 标签：AI 资源的多维元数据

标签（Tag）是一种键值对（Key-Value）形式的元数据，可以附加到几乎所有阿里云资源上。标签主要用于：

- 多维度资源分类（项目、环境、负责人等）
- 自动化运维（基于标签触发弹性伸缩、数据备份等）
- 成本分析（按标签维度生成费用报表）

(1) 标签体系设计

以下是常见的 AI 资源标签类型，仅供参考

标签 Key	可选 Value	说明
ai/project	fraud-detection,customer-service-bot	所属 AI 项目
ai/environment	dev,staging,prod,poc	环境标识
ai/workspace	ws-fd-prod,ws-md-dev	所属 PAI 或者百炼工作空间
ai/sensitivity	public,internal,confidential	数据敏感等级
ai/cost-center	dept-marketing,dept-finance	成本归属部门

所有新建 AI 资源必须强制打标，建立标签的巡检机制，及时发现没有绑定标签的云产品并评估影响和制定应对策略。

特殊限制

阿里云支持标签的产品

1.5. 总结

合理的资源规划是 AI Landing Zone 成功的基石。本文提出的方案旨在帮助您构建一个兼具性能、成本和安全性的 AI 基础设施。

核心建议是：

- **以多账号为基，实现硬隔离：**采用基于资源目录的多账号架构，从根源上实现业务、环境和安全责任的清晰分离，是构建安全合规 AI 平台的第一步。

- **因地制宜，规划平台资源：**根据 AI 业务所处的平台层级（MaaS/PaaS/IaaS），采用相匹配的资源组织方式，如工作空间、资源配额、集群规划等，实现资源在隔离与共享之间的平衡。
- **统一治理，实现精细化管理：**全面推行以资源组和标签为核心的治理策略，将每一份云上资源都纳入统一的成本分摊、权限控制和自动化管理范畴，实现精细化运营。

通过遵循这些原则进行系统性规划，您可以避免常见的资源混乱和成本失控问题，为 AI 业务的长期发展和创新奠定一个安全、高效且经济的坚实基础。

2. 财务管理

2.1. 概述

财务管理是 AI Landing Zone 中不可或缺的核心能力，它旨在通过将财务治理能力前置嵌入运维流程，为高成本、高弹性的大模型应用建立基本秩序。本模块的核心价值在于通过事前预算、精准归因和多维度的成本追踪，实现“控风险、明责任、助决策”三大目标，为 AI 业务的稳健落地与可持续发展，构建一个可信、透明、可控的财务基础。

2.2. 背景与挑战

在企业大模型应用建设的早期阶段，财务管理面临高弹性、高成本、高不确定性的独特挑战。AI 资源的申请、调度与消费高度依赖工程团队，而财务视角往往滞后介入，造成技术与财务的“双轨运行”。因此，必须识别并应对以下核心挑战：

- **成本难预估：**缺乏历史数据与标准化模型，预算编制依赖粗略估算，易与实际支出严重偏离。
- **归属不清晰：**多团队共享基础设施，若未在资源创建时强制绑定业务主体，成本无法有效归因，形成“糊涂账”。
- **治理滞后：**财务管控常停留在月度账单复盘阶段，缺乏事前配额约束与事中监控能力，难以有效控制成本。
- **协同断层：**技术指标（如 GPU 小时、token 量）与财务语言（如项目、成本中心）脱节，技术与财务团队缺乏统一语言与协同机制。

这些挑战提示我们，必须将基础财务治理能力（如标签、配额、可见性）前置嵌入运维平台，才能为大模型应用的可持续发展筑牢底线。

2.3. 具体方案

为应对上述挑战，我们构建一套以“预算控制、成本归属、成本追踪”为核心的财务管理方案。

2.3.1. 预算控制

(1) 事前约束，而非事后追责

预算控制的核心并非“算清花了多少”，而是在资源申请阶段设定明确边界，防止超支。一次训练任务可能消耗数万元，若等到月度账单出具再干预，损失已无法挽回。因此，必须将控制点前移至“资源创建前”——所有 GPU 实例、存储空间、推理服务等关键资源的创建，必须关联有效预算配额，否则自动拒绝。

(2) 以项目/团队为单位，粒度适中

预算单元不宜过细（如按模型版本），也不宜过粗（如全公司一个池子）。推荐以业务项目或技术团队为最小预算主体，既能满足责任归属需求，又避免管理开销过大。在缺乏历史数据的初期，可通过合理配额建立基本约束，防止资源被个别团队垄断或滥用。

(3) 配额与预算联动，软硬结合

- **硬管控**：对关键资源（如卡数、存储容量）设置硬性上限；
- **软管控**：对总成本设置预警阈值（如 80%），触发通知但不中断服务，兼顾灵活性与风险控制。

(4) 优先引导使用预付费资源，提升预算可预测性

在满足业务需求的前提下，鼓励团队优先选择预付费资源（如 PAI 计算资源组）。尽管其灵活性低于后付费资源（如 PAI 公共资源组），无法完全按需伸缩，但因其成本固定、无突发账单风险，更利于预算编制与执行管控。对于训练任务或稳定推理服务，预付费是更优的治理选择。

2.3.2. 成本归属

(1) 以使用方为成本责任主体

成本归属不以“资源归属”为唯一依据，而是根据谁发起使用行为划分责任：

- **模型开发/训练阶段**：由资源申请者（或所属项目）承担全部成本，资源标签在创建时绑定；
- **模型推理阶段：**由调用方（如业务系统、应用服务）按实际调用量（如输入/输出 token 数）分摊成本，而非由模型所有者承担。

(2) 训练资源标签必须统一且一致

所有模型开发、训练任务、微调作业等操作，在提交时必须指定 project、team 等归属标签，并确保关联资源（如向量数据库、对象存储）的标签保持一致，避免归属割裂。

(3) 推理成本按调用方拆分，支持多模型混合调用

当一个业务请求调用多个模型（如先调用 Embedding 模型，再调用 LLM），平台需记录每个子调用的 token 消耗，并将总成本按比例拆分至各调用方（或统一归属至发起请求的顶层应用）。

> *示例：某客服系统一次对话调用 Model-A（1000tokens）和 Model-B（500tokens），则总推理成本按 2:1 拆分至该客服系统名下（或进一步拆至两个子服务）。*

这种“谁受益、谁承担”的机制，能有效提升团队成本意识，促使其主动优化 prompt、启用缓存或选择更经济的模型。

(4) 杜绝无标签资源，平台强制校验

及时发现任何未携带完整归属标签的资源（训练或推理）。即使前期不实施 Chargeback，清晰的调用链与归属数据也能为后续的模型服务定价、跨团队结算或 ROI 评估提供可靠基础，避免“历史数据不可用”的窘境。

2.3.3. 成本追踪

(1) 追踪到可解释的最小业务单元

成本数据不仅展示“花了多少钱”，还需关联可理解的业务行为，例如：

- **训练任务**：模型名称、训练时长、GPU 类型、数据集版本；
- **推理调用**：调用方应用、模型名称、输入/输出 token 数、请求时间、是否缓存命中；
- **存储费用**：绑定模型 ID 或项目名，标注存储类型（权重、日志、缓存）及保留周期。

确保每一笔费用都能回答：“为什么会产生这笔钱？”

(2) 保留原始计量数据，支持双向追溯

平台需同时保存两类数据：

- **原始计量数据**：如 GPU 小时数、token 消耗量、存储 GB·天；
- **转换后成本数据**：基于单价计算出的金额。

运营人员可从“金额”反查“用量”，财务人员可验证“单价×用量=总价”，实现高效对账。

(3) 与云账单对齐，支持外部核验

内部成本数据的粒度与时间范围应能与阿里云账单对齐。当出现差异时，可快速定位是平台计量偏差、标签缺失，还是资源未纳管。

(4) 提供多维下钻视图，满足不同角色需求

- 管理层：按项目/团队查看月度趋势；
- 运营人员：按模型/服务分析调用量与成本关系；
- 财务人员：按成本科目（如“AI-训练”“AI-推理”）导出对账明细。

所有视图共享同一套底层数据，确保口径一致。

(5) 保留历史快照，支持审计与回溯

成本数据每日自动归档，保留至少 12 个月。即使资源已释放、标签已变更，仍可还原任意时间点的成本构成，满足内审或合规要求。

> *只有知道"Model-A 的推理成本中 70%来自长输出"，才能针对性优化；只有知道"某项目 80%成本在 dev 环境"，才能推动资源清理。成本追踪是优化的前提。*

2.4. 总结

在 AI Landing Zone 中，财务管理的核心目标并非构建复杂的计费系统，而是通过轻量、前置、平台化的治理能力，为高成本、高弹性的 AI 资源使用建立基本秩序。围绕这一目标，本模块通过三大能力形成闭环：

- **预算控制：**从源头设限，以项目/团队为单位设定配额，优先引导使用预付费资源，确保投入“可控”。
- **成本归属：**以使用方为责任主体，区分训练（创建者承担）与推理（调用者按量分摊），支持多模型场景下的精准归因，确保责任“可溯”。
- **成本追踪：**提供可解释、可核对的成本明细，关联业务上下文与原始计量数据，支撑运营排查与财务对账，确保支出“可审”。

三者协同，共同实现“花得明白、控得住、对得清”的初期治理目标，确保大模型平台在快速迭代的同时，始终运行在可信、透明、可持续的轨道上。

3. 网络规划

3.1. 概述

网络规划是构建 AI Landing Zone 的基石，它如同设计一套贯穿全局的"神经网络"，为数据的高效流转、算力的无缝协同以及服务的安全交付提供基础保障。本章旨在提供一个贯穿 AI 业务全生命周期（数据采集、模型训练、模型推理）的网络设计蓝图。我们将从 VPC 划分、路由设计、跨域互联到安全隔离等多个维度，系统性地阐述如何构建一个高性能、高安全、高可用且具备成本效益的 AI 网络基础设施，确保您的 AI 应用在阿里云上稳定、高效地运行。

3.2. 背景与挑战

AI 业务对网络的要求远超传统应用，其独特的业务流程和数据密集型特点，给网络规划带来了四大核心挑战：

- **极致的性能要求**：尤其在模型训练阶段，大规模分布式训练任务需要在数百乃至数千个计算节点间进行高频的梯度同步，这要求网络具备超低延迟、超高吞吐和接近无损的通信能力（如 RDMA），传统 TCP/IP 网络难以满足。
- **复杂的连接场景**：AI 生命周期涉及与全球多点数据源（公网）、企业本地数据中心（IDC）、云上多地域 VPC 以及最终用户的广泛连接。如何构建一个安全、高效的混合云、多地域全球网络，是规划中的一大难题。
- **严苛的数据安全与合规**：AI 系统处理的数据量巨大，且往往包含商业机密或个人隐私等敏感信息。网络设计必须在保障数据高效传输的同时，实现严格的访问控制和逻辑隔离，防止数据在流转过程中被窃取或滥用，并满足不同地域的合规要求。
- **架构的弹性与扩展性**：AI 应用架构多样，从 MaaS、PaaS 到 IaaS，不同部署模式对网络的需求各异。同时，AI 业务负载（如突发的推理请求）具有高度不确定性。网络架构必须具备足够的灵活性和弹性，以支持业务的快速迭代和未来规模的平滑扩展。

3.3. 具体方案

AI 业务，按照过程分为 3 个阶段，每个阶段分别对网络提出了不同的能力要求：

- **数据采集与预处理**：就近数据源端提升数据采集效率并节约采集流量成本、全球内网支撑数据回传和聚集速度、支持多种数据标注工具安全接入
- **模型训练**：构建全球、多云统一算力资源池、提供高吞吐数据通道，保障低延迟、无损和安全通信，提升训练效率
- **AI 推理**：就近接入推理、全球内网智能调度调用，实现毫秒级时延闭环和安全应用

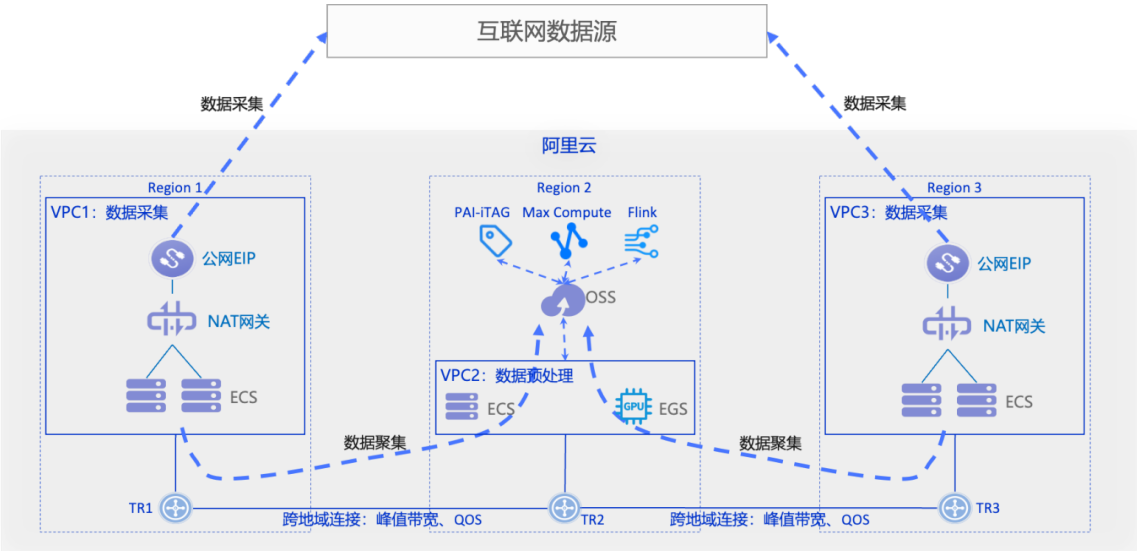
参考《AI LandingZone 白皮书》中资源规划章节进行整个企业的 AI 资源和已有业务的规划。假设资源规划和多账号体系设计已经完成，本章节阐述账号内部的网络规划，即：

- 针对 3 个阶段的网络资源划分 VPC
- 规划设计 VPC 内网络
- 规划设计 VPC 和 VPC 间、VPC 和公网、VPC 和 IDC 间等的互通网络（要考虑已有业务的 VPC）

3.3.1. 数据采集与预处理阶段

全球训练数据聚集与预处理网络方案的核心在于提供大规模、弹性的公网 IP 与带宽资源，支撑从全球互联网高效采集海量原始训练数据。依托阿里云全球基础设施，方案可按需提供大量独立公网 IP 和高带宽出口，有效应对采集过程中的 IP 问题、并发限制和带宽瓶颈，显著提升数据获取的并发能力与成功率。采集完成后，方案无缝衔接跨地域高速传输能力，通过阿里云全球网络实现数据在不同区域之间的低延迟、高吞吐汇聚，确保分散采集的数据能快速、稳定地集中至指定计算节点进行预处理。方案不仅保障

公网采集端的资源供给，也打通了全球多地域间的数据流转通道，实现“采集–传输–预处理”全链路高效协同。整体架构以资源弹性、全球部署和跨域互联为优势，为大模型训练构建端到端的高性能数据管道。



3.3.2. VPC 划分

云上数据采集场景的 VPC 划分应遵循最小权限、分层隔离、安全可控、弹性可扩展原则

可采用多 VPC 架构或者单 VPC 多子网架构。

(1) 按业务功能分层划分

将数据采集、预处理、存储、训练和推理等不同阶段部署在不同的子网或 VPC 中，形成逻辑隔离。例如：

- 数据采集入口建议放在 DMZ VPC 或者公共子网；
- 预处理与中间数据暂存放在独立的处理 VPC；
- 敏感原始数据存储与 AI 训练放在高安全 VPC，禁止公网直接访问。

(2) 按安全等级隔离

根据数据敏感性和合规要求，划分高、中、低安全等级 VPC 或者高、中、低等级安全子网，并通过安全组、网络 ACL、私网连接控制跨 VPC\跨子网通信权限，禁止高敏感系统直接暴露公网。

(3) 按租户或项目隔离

在平台型数据采集系统中，可为不同租户或项目分配独立 VPC，实现资源、网络与数据的完全隔离，避免"邻居干扰"。

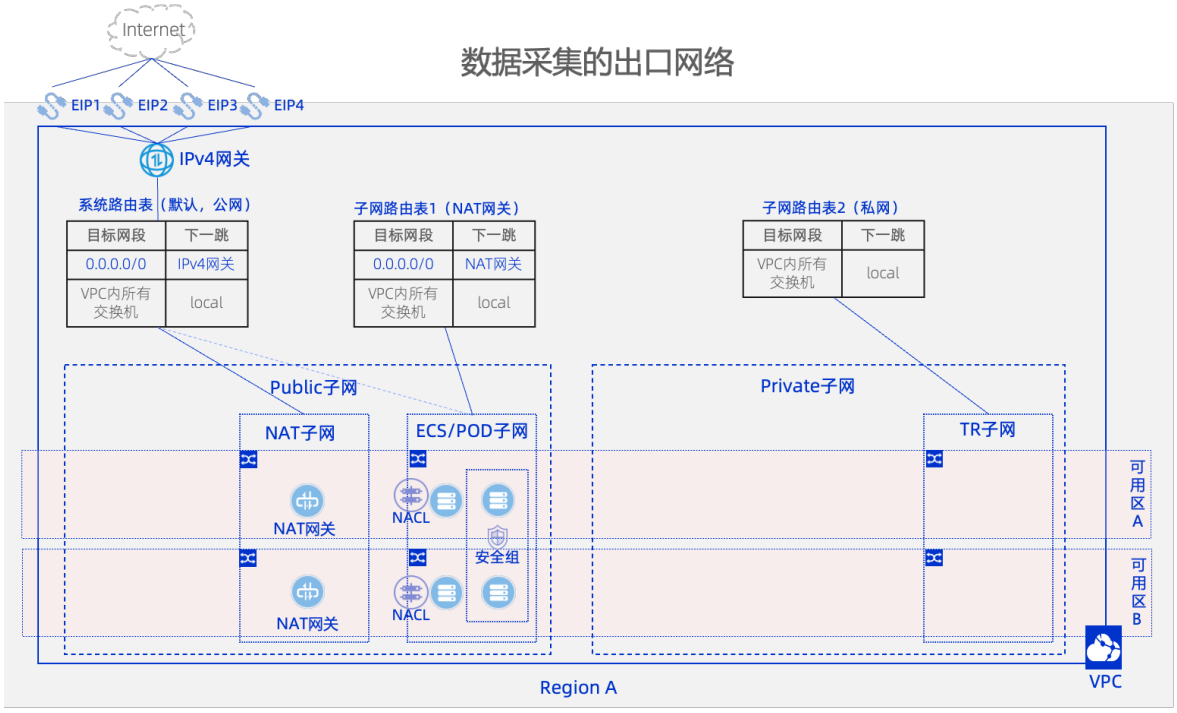
3.3.3. 数据采集出口网络（NAT 网关/自建代理方案）

- 通过弹性公网 IP（EIP）与共享带宽，灵活、经济地接入公网数据源，并统一管理出口带宽资源，进行数据汇聚。
- 基于云企业网（CEN）的网络全球互联领域，打通全球多个地域的 VPC，实现跨区域、低延迟、高带宽的私网互联，构建统一的数据传输网。
- NAT 网关保障私有网络中的计算节点安全访问互联网，实现数据采集与预处理的合规隔离；
- IPv6 网关则支持新一代网络协议，用于进行 IPv6 资源数据聚集。

整套网络方案在保障数据传输安全、稳定的同时，显著提升全球数据汇聚效率与预处理性能，为大规模 AI 训练提供坚实、敏捷的网络基础设施支撑。

NAT 网关方案：使用 EIP 地址池，提升公网 IP 数量；使用 NAT 网关作为出口，采集程序 SNAT 随机公网 IP 出向访问。适用于通用数据采集场景

自建代理方案：客户自建代理机器，维护数据采集进程和公网 IP 之间的映射关系，把同一个数据采集请求分拆到不同的代理机器上出公网。

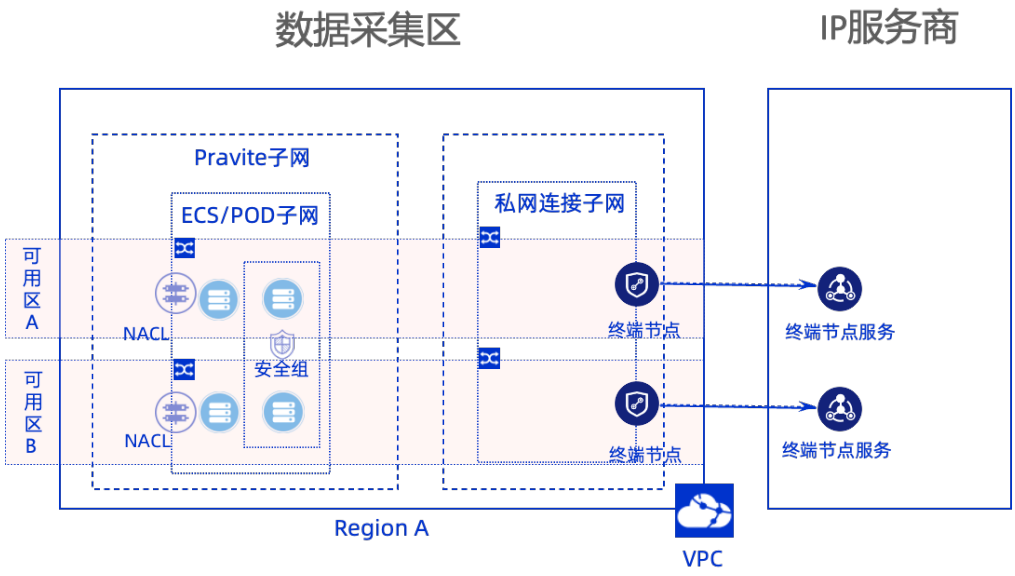


就近数据源地域选定 Region 部署 VPC，VPC 内部署 NAT 网关绑定 EIP，配置 SNAT 规则供 VPC 内 ECS 或容器 POD 做出向访问、采集数据。如果需要海量公网 IP 作为采集源 IP 时，可以考虑使用 Private Link 集成三方代理 IP 方案。

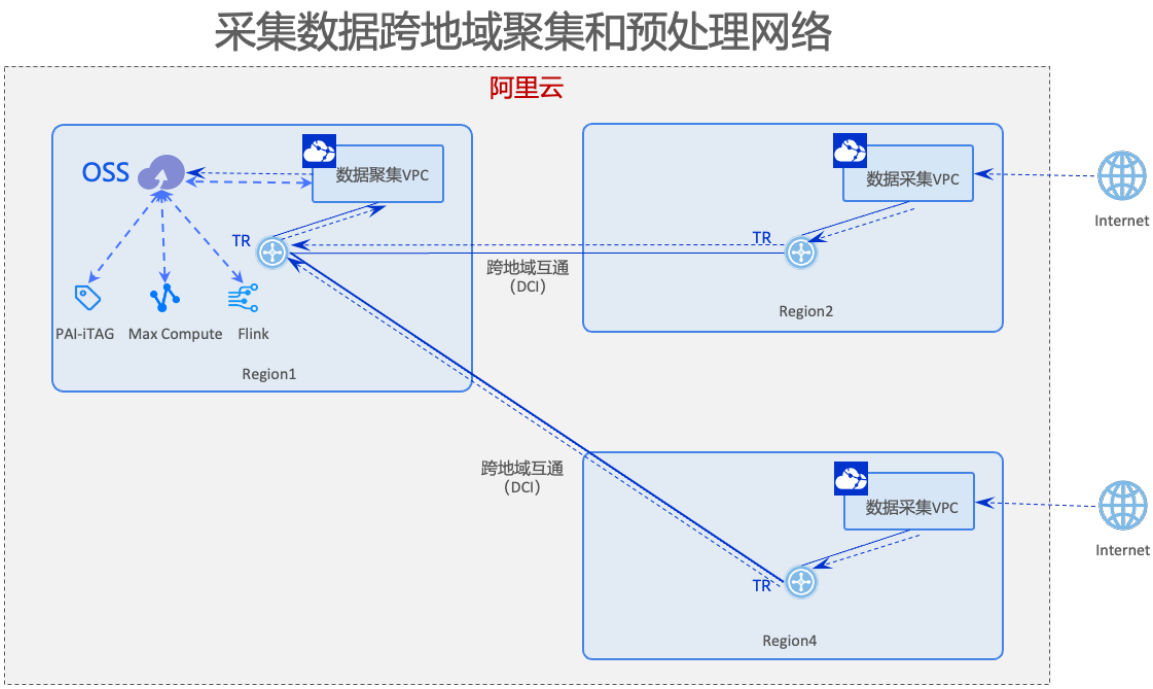
- VPC 详细设计参考：同地域单 VPC 网络设计
- 出站详细设计可以参考：同 VPC 内多公网 NAT 网关部署方案

3.3.4. 数据采集出口网络（IP 服务方案）

阿里云全球训练数据采集网络方案协同 IP 服务商，提供大规模、弹性的公网 IP 与带宽资源，支撑从全球互联网高效采集海量原始训练数据。依托阿里云全球基础设施，方案使用阿里云私网连接产品通过云上内网访问 IP 服务商，具备访问成本低，安全可控，质量稳定等优势，并通过 IP 服务商完成数据采集工作。



3.3.5. 采集数据跨地域聚集和预处理网络



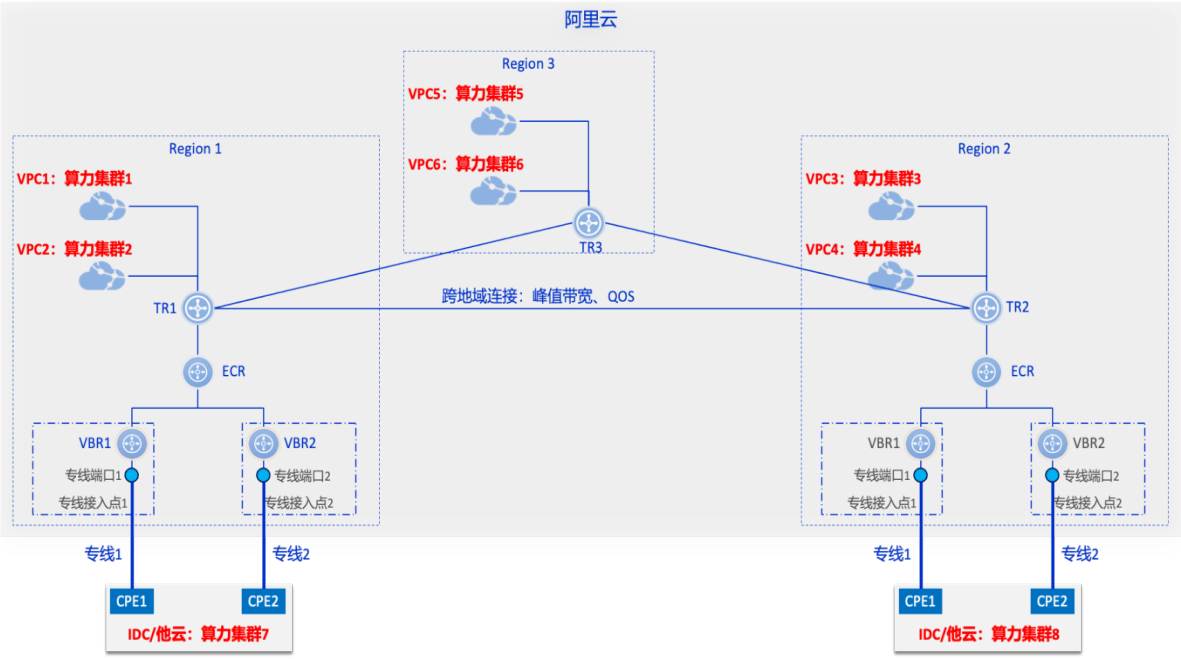
(1) 根据数据预处理集群规划，就近选定存放数据的 OSS 所在 Region、在同 Region 内创建数据聚集 VPC（此 VPC 内暂时不需要创建资源，只是通过它中转访问 OSS）；各地域创建 TR、连接本地 VPC 并 TR 间互联，打通各地域数据采集 VPC 可以直接访问数据聚集地 OSS 的网络通道。

- 跨地域多 VPC 互通详细设计可以参考：CEN 构建云上跨地域网络
- 访问 OSS 配置参考：配置访问云服务

(2) 在 OSS 所在地域创建 PAI-iTAG、Max Compute、Flink 等云服务，将 OSS 内采集到的数据进行数据标注等预处理后，再写回 OSS。也可以在数据采集 VPC 内创建 CPU 或 GPU 算力、自己部署工具进行数据预处理。

3.4. 模型训练阶段

模型训练的网络和数据采集与数据预处理的网络紧密相连：训练网络消费采集后经过预处理的海量数据集，因此它们共同构建了一张全球网络。这张网络不仅跨越地域边界，连接云上数据中心、边缘节点与本地 IDC，还通过高速互联（如阿里云 CEN、TR、GA 等）实现低延迟、高吞吐的数据流转。数据从源头（如 IoT 设备、用户行为日志、公开数据集）被采集后，经由预处理流水线（包括清洗、标注、格式转换、特征工程等）写入高性能存储系统（如 CPFS、OSS 或 NAS），再由训练集群按需拉取。为保障端到端效率，整个链路需在带宽、安全策略、访问控制和容灾能力上协同设计——例如，预处理任务可部署在靠近数据源的区域以减少传输开销，而训练任务则调度至具备 RDMA 加速能力的灵骏集群；同时，通过统一的 VPC 规划和路由策略，确保数据在采集、预处理与训练各阶段无缝、安全、高效地流动。这种一体化的网络与数据架构，已成为支撑大模型时代 AI 工程化落地的核心基础设施。本章介绍训练网络的设计。

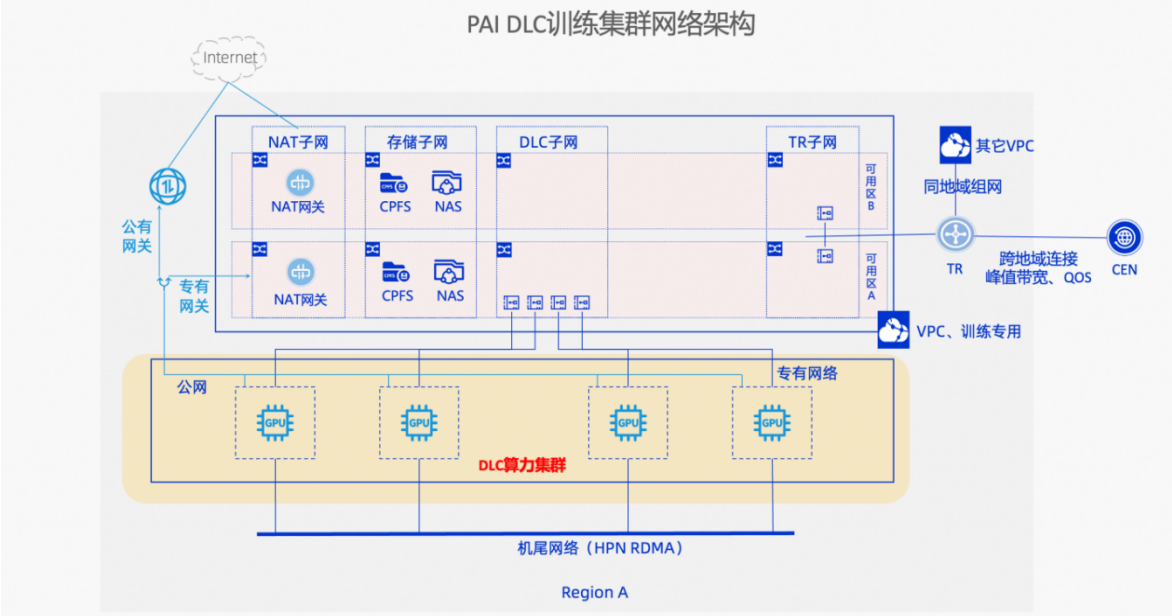


3.4.1. 基于 PAI DLC 训练的网络规划与设计

DLC 是由阿里云人工智能平台 PAI 提供的训练平台，它帮助用户快捷地创建单机或分布式训练任务，适用于需要快速启动训练任务的用户，支持多种深度学习框架，并提供灵活的资源配置选项。它的底层使

用 Kubernetes 拉起计算节点，对用户屏蔽了大部分关于集群、网络、存储的细节，避免了用户手动购买机器并配置运行环境，且无需改变原有使用习惯。

(1) 网络总览



对于 DLC 算力集群，每个 worker 节点接入以下三个网络：

- 机尾网络：机尾网络是支持 RDMA 的高性能网络，用于节点间的高速通信，特别是在分布式训练中传输梯度、模型参数等。
- 专有网络：用于连接用户的私有网络，比如云上同地域或跨地域的 VPC、线下 IDC。这个连接是可选的：
    - 用户没有连接到专有网络。每个算力 worker 节点仍然会连接到一个共享的 VPC，该网络只用于拉取镜像、访问 OSS，无法连接用户的私有网络。
    - 用户连接到自己的某个 VPC，通过该 VPC 可以连接用户的私有网络，比如云上同地域或跨地域的 VPC、线下 IDC。当使用 CPFS 作为数据集源、或者使用灵骏竞价资源时，都必须使用 VPC。
- 公网：仅支持由内到外的主动访问。DLC 提供两种公网访问方案：
    - 公有网关，所有的用户共享公网带宽，在用户并发高时下载速率会较慢，可能导致训练任务下载外部数据集较慢。
    - 专有网关，独享带宽，可根据需求配置带宽，仅当用户选择连接到专有网络模式下可以选择使用。使用专有网关务必按照操作手册配置好访问路由，并在启动后的实例中验证配置是否成功。

基于 Landing Zone 最佳实践，推荐如下配置：

- 规划 VPC 作为 DLC 算力集群的专有网络，并且将存放训练数据集的 CPFS/NAS 也放在该 VPC 下。
- 使用专有网关模式，提升 worker 节点访问公网的速度和稳定性。

(2) 网络设计

确定地理位置

- 根据云资源尤其是灵骏智算资源、CPFS 的储备情况、可用区时延情况选择可用区 AZ

训练专用 VPC 设计

划分训练专用 VPC，用来存放训练用到的数据，以及 worker 节点访问公网的出口。建议该 VPC 不对外提供服务，不部署 SLB 等公网暴露型资源

建议该 VPC 预留创建用于部署 ACK 灵峻集群的 POD 子网，参考 2.3 节

- **子网网段设计**
    - NAT 子网：一般情况下，每个可用区规划一个/28 网段可以满足需求，最大支持放置 12 个 NAT 实例。
    - TR 子网：一般情况下，每个可用区规划一个/28 网段可以满足需求，最大支持连接到 12 个 TR。
    - 存储子网：根据数据集的规模和划分确定需要多少 CPFS/NAS 实例，每个实例在每个可用区占用一个 IP。一般情况下，每个可用区规划一个/28 网段可以满足需求。
    - DLC 子网：根据需要的 worker 节点数量确定网段，每个 worker 节点占用一个 IP。
- **VPC CIDR 规划**

  VPC CIDR 应该覆盖上述子网网段，并预留一定的扩容空间后，选择覆盖上述空间的最小网段作为 VPC 的 CIDR，且应避免与用户其他业务网段发生冲突。**不要选择 10.0.0.0/8 等大段，**否则可能导致后续无法复用该 VPC 支持 ACK 灵骏集群训练场景
- **路由表设计**
    - 系统路由表（默认）：1）创建 VPC 后，系统会默认创建一张系统路由表，此表有到 VPC 内所有交换机的路由；2）在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 IPv4 网关。该路由表允许 VPC 内访问，和所有绑定 EIP 的资源（ECS/NAT 网关）访问公网。
    - 子网路由表 1（出公网）：1）在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 NAT 网关。该路由表允许 VPC 内访问，以及通过 NAT 网关访问公网。
    - 子网路由表 2（私网）：1）在此表中添加静态三大段 10.0.0.0/8、172.16.0.0/12、192.168.0.0/16 指向 TR attachment。该路由表允许所有私网访问。

- 子网路由表 3（私网+公网）：1）在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 NAT 网关。2）在此表中添加静态三大段 10.0.0.0/8、172.16.0.0/12、192.168.0.0/16 指向 TR attachment。该路由表允许所有私网和公网访问。

- 子网路由表 4（VPC 内）：1）创建该表后，不添加任何自定义路由。该路由表仅允许 VPC 内访问。

- 子网（交换机）和路由表关联关系

  - NAT 子网：关联到系统路由表。

  - TR 子网：

    - 该 VPC 作为组网中的统一公网出口，则关联到子网路由表 1（出公网）。
    - 其它情况下，关联到子网路由表 4（VPC 内）。

  - 存储子网

    - 关联到子网路由表 2（私网），允许数据采集程序通过 CEN 和 TR 跨 VPC 写入采集数据，读取训练的结果等。

  - DLC 算力子网：

    - DLC 训练集群需要到公网拉取数据，并且也需要跨 TR 访问其它资源，则关联子网路由表 3（私网+公网）。
    - DLC 训练集群仅需要到公网拉取数据，则关联子网路由表 1（出公网）。
    - DLC 训练集群不需要访问公网，但需要跨 TR 访问其它资源，则关联子网路由表 2（私网）。
    - DLC 训练集群不需要访问公网，也不需要跨 TR 访问其它资源，则子网路由表 4（VPC 内）。

- **安全防护设计（可选）**

  - 网络 ACL（NACL）可以控制交换机（子网）粒度流量阻断，网络 ACL 是无状态的。不建议在 TR 子网/NAT 子网上配置 NACL。

  - 在创建 DLC 算力集群时，需要指定 Worker 节点接入私网用的安全组，并且只能指定唯一一个安全组。建议为每个 AZ 的算力集群创建专用的安全组规则，并按需配置安全组规则。

- **公网网关设计（可选&推荐）**

  - 推荐使用专有网关模式，提升 worker 节点访问公网的速度和稳定性。

(3) 场景指南

通过专有网关提升公网访问速率

由于 DLC 默认使用共享网关，受带宽限制，下载大型文件时网速可能无法满足需求。因此，当您想要提升网络上传和下载速度时，可以为实例所在专有网络（VPC）创建公网 NAT 网关、绑定弹性 IP（EIP）并配置 SNAT，从而使实例通过专有公网网关高速访问互联网。参考通过专有网关提升公网访问速率

worker 节点访问海外公网加速

当在训练中使用到海外的数据，如 Hugging Face 的数据集，或者是海外的镜像时，可以使用 GA 进行访问公网加速。请参考：跨域拉取海外模型或容器镜像

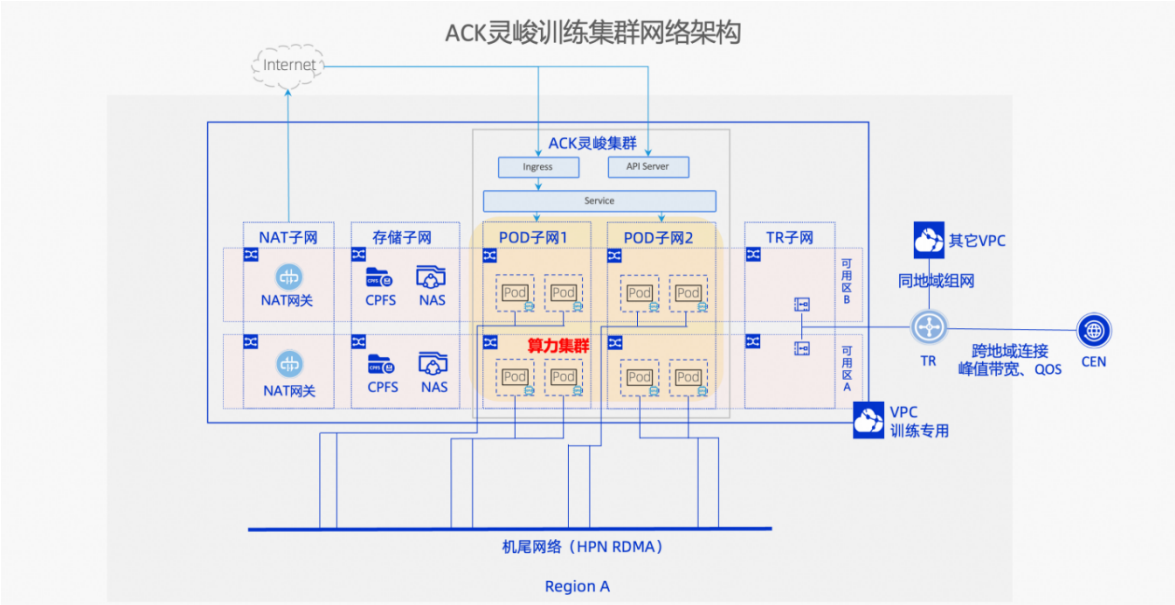
worker 节点不允许访问公网

参考前一节的子网（交换机）和路由表关联关系，将 DLC 算力子网关联到子网路由表 2（私网）或者子网路由表 4（VPC 内）。

3.4.2. 基于 ACK 灵骏训练的网络规划与设计

阿里云容器服务 ACK 灵骏托管版集群是容器服务 Kubernetes 版（ACK）针对智能计算灵骏提供的集群类型，提供全托管和高可用控制面板的标准 Kubernetes 集群服务，支持以灵骏计算节点作为 Kubernetes 集群的工作节点。通过对基础设施资源的深度管理，通过 ACK 灵骏集群能够实施 GPU 直连、NUMA 绑定、多网卡多队列优化、自定义调度策略等，有利于充分发挥超大规模 GPU 集群的性能潜力。

3.5. 网络总览



对于 ACK 灵骏算力集群，它的网络插件只支持 Terway 插件，它的网络是由容器网络和机尾网络组成：

- POD 网络，容器网络的一部分，通过 Terway 网络插件接入 VPC，并通过 VPC 和 TR 连接用户其它的私有网络和公网，比如云上同地域或跨地域的 VPC、线下 IDC。

- Service 网络，容器网络的一部分，由 Kubernetes 原生的 Service 机制实现，基于 kube-proxy 和 iptables/IPVS 规则提供集群内部的服务发现与负载均衡能力。在 ACK 灵骏集群中，Service 的 ClusterIP 地址段独立于 VPC 网段。
- 机尾网络。机尾网络是支持 RDMA 的高性能网络，用于节点间的高速通信，特别是在分布式训练中传输梯度、模型参数等。

(1) 网络设计

确定地理位置

根据云资源尤其是灵骏智算资源、CPFS 的储备情况、可用区时延情况选择可用区 AZ

训练专用 VPC 设计

划分训练专用 VPC，用来存放训练用到的数据，以及 ACK 灵骏算力集群节点访问公网的出口。建议该 VPC 不对外提供服务，不部署 SLB 等公网暴露型资源

建议该 VPC 预留创建用于部署 DLC 训练集群的子网，参考 2.3 节

- **子网网段设计**
    - NAT 子网：一般情况下，每个可用区规划一个/28 网段可以满足需求，最大支持放置 12 个 NAT 实例。
    - TR 子网：一般情况下，每个可用区规划一个/28 网段可以满足需求，最大支持连接到 12 个 TR。
    - 存储子网：根据数据集的规模和划分确定需要多少 CPFS/NAS 实例，每个实例在每个可用区占用一个 IP。一般情况下，每个可用区规划一个/28 网段可以满足需求。
    - POD 子网：根据需要的 POD 节点数量确定网段，每个 POD 节点占用一个 IP。
- **VPC CIDR 规划**

  VPC CIDR 应该覆盖上述子网网段，并预留一定的库容空间后，选择覆盖上述空间的最小网段作为 VPC 的 CIDR，且应避免与用户其他业务网段发生冲突。**不要选择 10.0.0.0/8 等大段**，否则可能会因为 Service 网段跟 VPC CIDR 冲突无法创建集群。
- **路由表设计**
    - 系统路由表（默认）：

(1) 创建 VPC 后，系统会默认创建一张系统路由表，此表有到 VPC 内所有交换机的路由；

(2) 在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 IPv4 网关。该路由表允许 VPC 内访问，和所有绑定 EIP 的资源（ECS/NAT 网关）访问公网。

- 子网路由表 1（出公网）：在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 NAT 网关。该路由表允许 VPC 内访问，以及通过 NAT 网关访问公网。
- 子网路由表 2（私网）：在此表中添加静态三大段 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 指向 TR attachment。该路由表允许所有私网访问。
- 子网路由表 3（私网+公网）：

(1) 在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 NAT 网关。

(2) 在此表中添加静态三大段 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 指向 TR attachment。该路由表允许所有私网和公网访问。

- 子网路由表 4（VPC 内）：创建该表后，不添加任何自定义路由。该路由表仅允许 VPC 内访问。

- **子网（交换机）和路由表关联关系**
  - NAT 子网：关联到系统路由表。
  - TR 子网：
    - 该 VPC 作为组网中的统一公网出口，则关联到子网路由表 1（出公网）。
    - 其它情况下，关联到子网路由表 4（VPC 内）。
  - 存储子网：关联到子网路由表 2（私网），允许数据采集程序通过 CEN 和 TR 跨 VPC 写入采集数据，读取训练的结果等。
  - POD 子网：
    - ACK 训练集群需要到公网拉取数据，并且也需要跨 TR 访问其它资源，则关联子网路由表 3（私网+公网）。
    - ACK 训练集群仅需要到公网拉取数据，则关联子网路由表 1（出公网）。
    - ACK 训练集群不需要访问公网，但需要跨 TR 访问其它资源，则关联子网路由表 2（私网）。
    - ACK 训练集群不需要访问公网，也不需要跨 TR 访问其它资源，则子网路由表 4（VPC 内）。
- **安全防护设计（可选）**
  - 网络 ACL（NACL）可以控制交换机（子网）粒度流量阻断，网络 ACL 是无状态的。不建议在 TR 子网/NAT 子网上配置 NACL。

ACK 灵骏集群网络设计

- Service 网段规划：根据使用的 Service 数量规划网段，该网段不能与 VPC CIDR 重叠冲突。
- 是否对公网暴露 ACK 的 API service：根据业务需要选择。
- Ingress：根据业务需要选择：公网、私网、不需要。

公网网关设计【可选&推荐】

- 当训练业务需要访问公网，如从 Hugging Face 上拉取数据集时，请将 POD 子网关联到子网路由表 1（出公网）。

(2) 场景指南

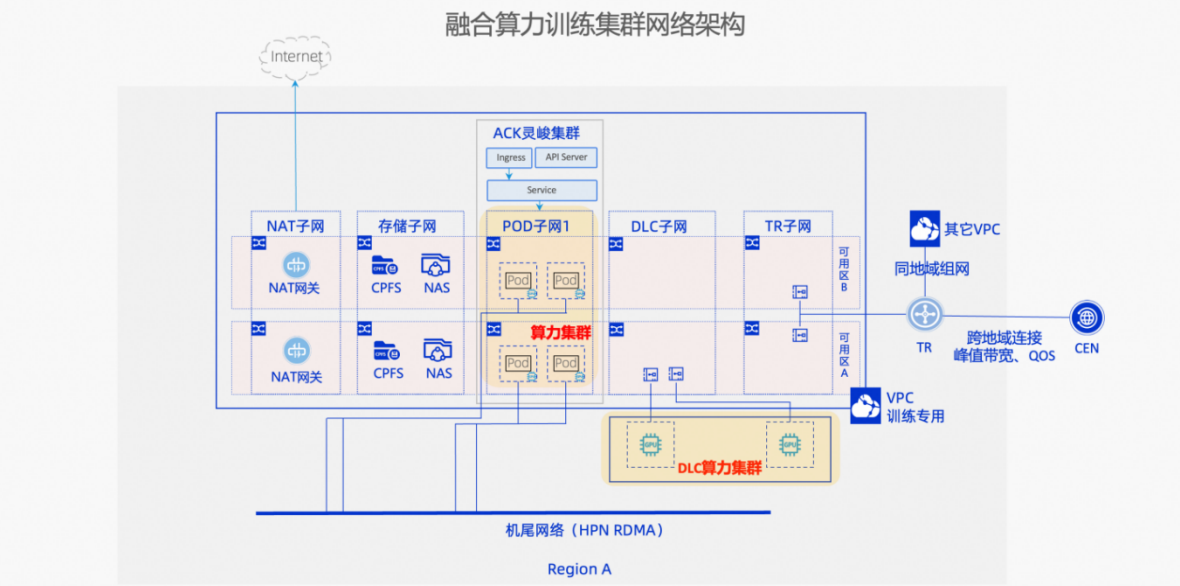
灵骏节点访问海外公网加速

当在训练中使用到海外的数据，如 Hugging Face 的数据集，或者是海外的镜像时，可以使用 GA 进行访问公网加速。请参考：跨域拉取海外模型或容器镜像

灵骏节点不允许访问公网

参考前一节的子网（交换机）和路由表关联关系，将 POD 子网关联到子网路由表 2（私网）或者子网路由表 4（VPC 内）。

3.5.1. 融合算力训练的网络规划与设计



在上面设计的基础上，很容易将训练网络扩展为既支持基于 PAI DLC 训练场景，又支持 ACK 灵骏集群训练场景的融合场景：

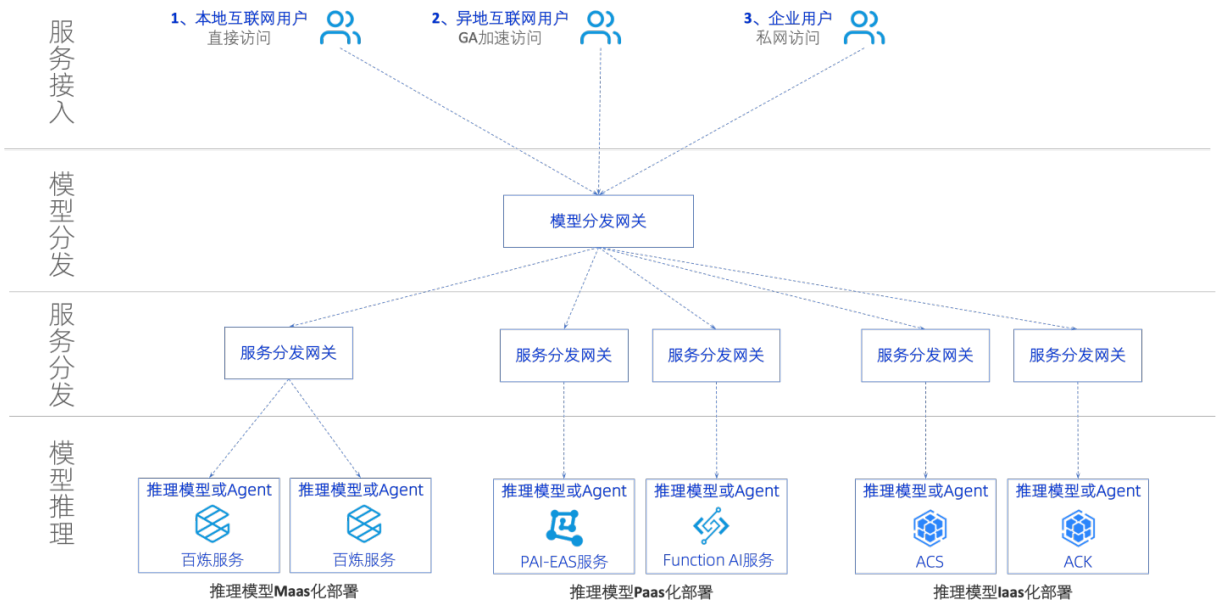
- 训练专用 VPC 中复用 NAT 子网、存储子网、TR 子网，共享相同的公网隔离和加速方案
- 分别为 ACK 灵峻集群和 DLC 算力集群规划相应的子网
- ACK 灵峻集群中的网络设计，请参考 2.2 节

- DLC 算力集群中的网络设计，请参考 2.1 节

注意：DLC 算力集群的 worker 节点也会连接机尾网络（图略）。

3.6. 模型推理阶段

从网络视角看大模型推理服务，自上而下可以分为 4 层：



- **服务接入层**：部署公网或私网接入网关，接入互联网用户或云上企业用户的推理请求
- **模型分发层**：部署模型分发网关，根据将推理请求分发到不同的后端模型，实际部署时可能省略
- **服务分发层**：部署模型对应的服务分发网关，根据推理请求信息将请求路由或负载到对应的模型服务上
- **模型推理层**：根据大模型推理业务特征、开发和运维能力等选择合适平台进行的模型部署

接下来将分层介绍各层的网络规划与设计。

3.6.1. 模型推理层的网络规划与设计

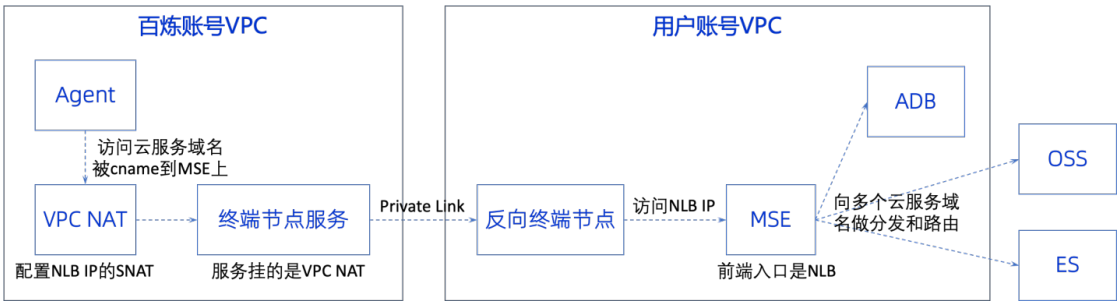
(1) 百炼平台 Maas 化部署模型的网络

阿里云的大模型服务平台百炼，是一站式的大模型开发及应用构建平台。

用户使用百炼平台部署模型或 Agent 时，实际是部署在百炼平台的 VPC 中，这个 VPC 被托管在百炼后台账号中，用户不可见、所以不需要规划和设计。

但是，百炼平台 VPC 和外界交互通信网络，需要用户了解和必要的配置：

- **百炼平台部署的模型/Agent 被互联网访问**：百炼平台后台隐藏部署了公网网关，用户可以直接使用公网域名发起 API 调用。具体参考：获取与配置 API Key。如果想以加密方式传输推理请求信息可以参考：以加密的方式接入模型推理功能
- **百炼平台部署的 Agent 访问互联网**：百炼平台后台隐藏部署了访问互联网的能力。具体参考：联网搜索
- **百炼平台部署的模型/Agent 被用户 VPC 私网访问**：用户 VPC 可以通过私网连接 PrivateLink 产品打通和百炼平台之间的私有网络，然后使用私网域名发起 API 调用。具体参考：私网访问阿里云百炼模型或应用 API
- **百炼平台部署的 Agent 私网访问用户 VPC**：百炼平台可以通过私网连接 PrivateLink 产品打通和用户 VPC 之间的私有网络，然后使用私网域名发起对用户账号的云服务调用。具体参考：安全存储，因为这个逻辑比较复杂，下面画图解释一下原理：



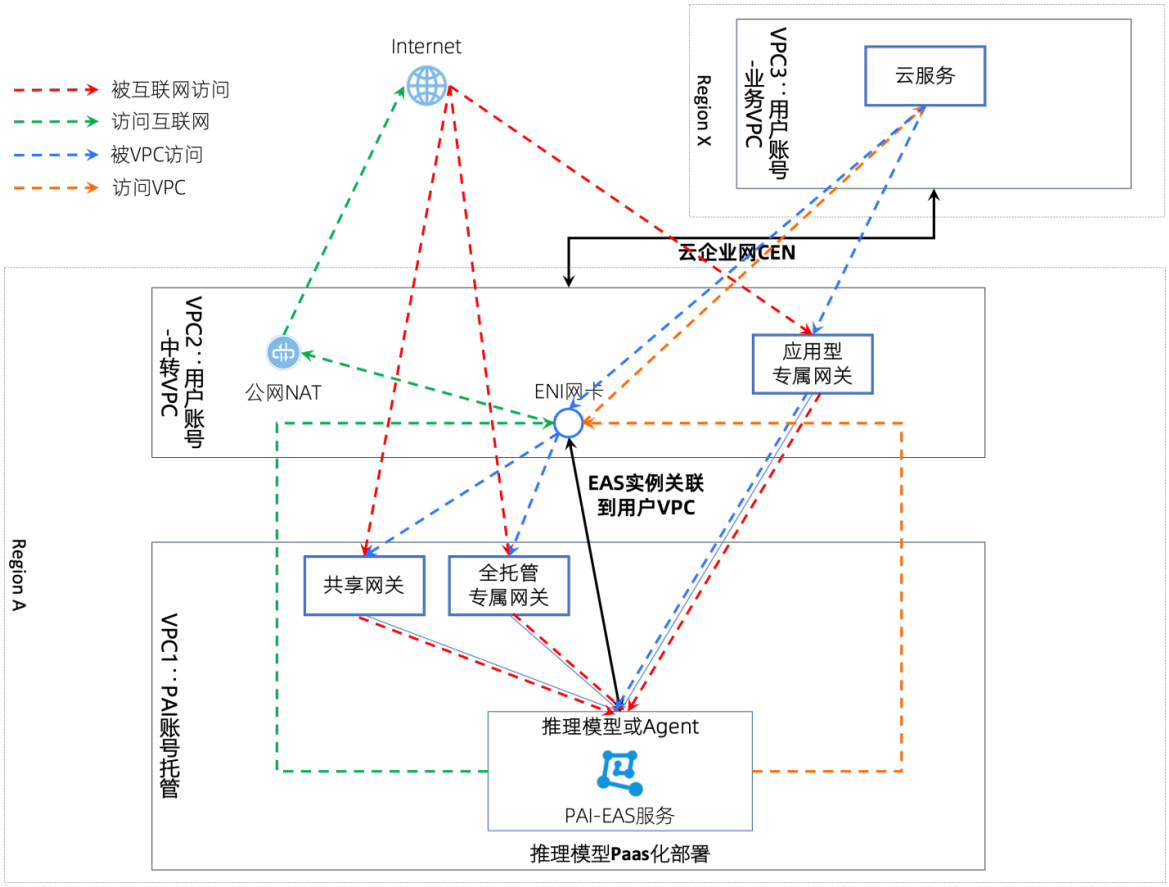
  - 在用户 VPC 内创建私网连接 PrivateLink 的反向终端节点后，终端节点服务侧可以通过这个反向终端节点访问到 VPC 内的私网 IP 地址（比如图中 MSE 网关内部集成的 NLB 私网 VIP、进而访问到 MSE，然后由 MSE 分发访问请求到 ADB、OSS、ES 等云产品）
  - 在百炼账号 VPC 内创建私网连接 PrivateLink 的终端节点服务、并关联 VPC NAT 网关后，百炼账号 VPC 内发起的对用户 VPC 的访问就会被 VPC NAT 网关转发到终端节点服务、进而转发到用户 VPC 的反向终端节点上
  - 百炼平台能访问用户 VPC 的哪些资源或云服务，由百炼平台控制（即：VPC NAT 网关是否配置了这些资源或服务 IP 的 SNAT 规则）
  - 此 PrivateLink 连接为单向设计，仅允许百炼平台的业务空间访问用户 VPC、用户 VPC 无法通过此连接访问百炼平台

(2) PAI-EAS 平台 Paas 化部署模型的网络

模型在线服务 PAI-EAS（Elastic Algorithm Service）是模型在线服务平台，支持用户将模型一键部署为在线推理服务或 AI-Web 应用。PAI-EAS 适用于实时推理、近实时异步推理等多种 AI 推理场景，具备自动扩缩容和完整运维监控体系等能力。

用户使用 PAI-EAS 平台部署模型或 Agent 时，实际上是部署在 PAI 平台的 VPC 中，这个 VPC 被托管在 PAI 后台账号中，用户不可见、所以不需要规划和设计。

但是，PAI-EAS 平台 VPC 和外界通信网络，需要用户了解并进行必要的配置：



- **PAI-EAS 平台部署的模型/Agent 被互联网访问**：如上图中红色虚线箭头所示，有 3 种方式被互联网（公网访问）：
  - **共享网关方式**：在模型部署阶段，默认会创建共享网关（托管在 PAI-EAS 平台 VPC 中），共享网关上提供了公网域名供互联网用户访问。具体参见：通过网关进行公网或内网调用（默认）
  - **全托管专属网关方式**：用户可以自行创建全托管网关（托管在 PAI-EAS 平台 VPC 中）、在网关上开启被公网访问开关、添加公网 IP 白名单后关联到 EAS 实例。具体参见：1.3 配置自定义域名
  - **应用型专属网关方式【推荐】**：用户可以自行创建全托管网关（生成在用户 VPC 中、被 EAS 管理）、在网关上开启被公网访问配置后关联到 EAS 实例。具体参见：1.3 配置自定义域名
- **PAI-EAS 平台部署的 Agent 访问互联网**：如上图绿色虚线箭头所示，PAI-EAS 实例需要先关联到用户 VPC（在用户 VPC 内插入 ENI 网卡）、用 VPC 内创建公网 NAT 网关并配置 SNAT，借助用户 VPC 出公网。具体参见：网络配置
- **PAI-EAS 平台部署的模型/Agent 被用户 VPC 私网访问**：如上图蓝色虚线箭头所示，有 3 种方式被用户 VPC 私网访问：

- **共享网关和全托管网关方式**，PAI-EAS 实例需要先关联到用户 VPC（在用户 VPC 内插入 ENI 网卡），然后此 VPC 可以通过共享网关和全托管网关的私网域名访问，用户的其他 VPC 可以先通过 CEN 连通到此 VPC 再访问网关私网域名。具体参见：通过网关进行公网或内网调用，使用 EAS 专属网关实现访问控制和自定义域名调用

- **应用型专属网关方式，**因为应用型专属网关就生成在用户 VPC 中，所以此 VPC 可以直接访问应用型专属网关的私网域名，用户的其他 VPC 可以先通过 CEN 连通到此 VPC 再访问网关私网域名。具体参见：使用 EAS 专属网关实现访问控制和自定义域名调用

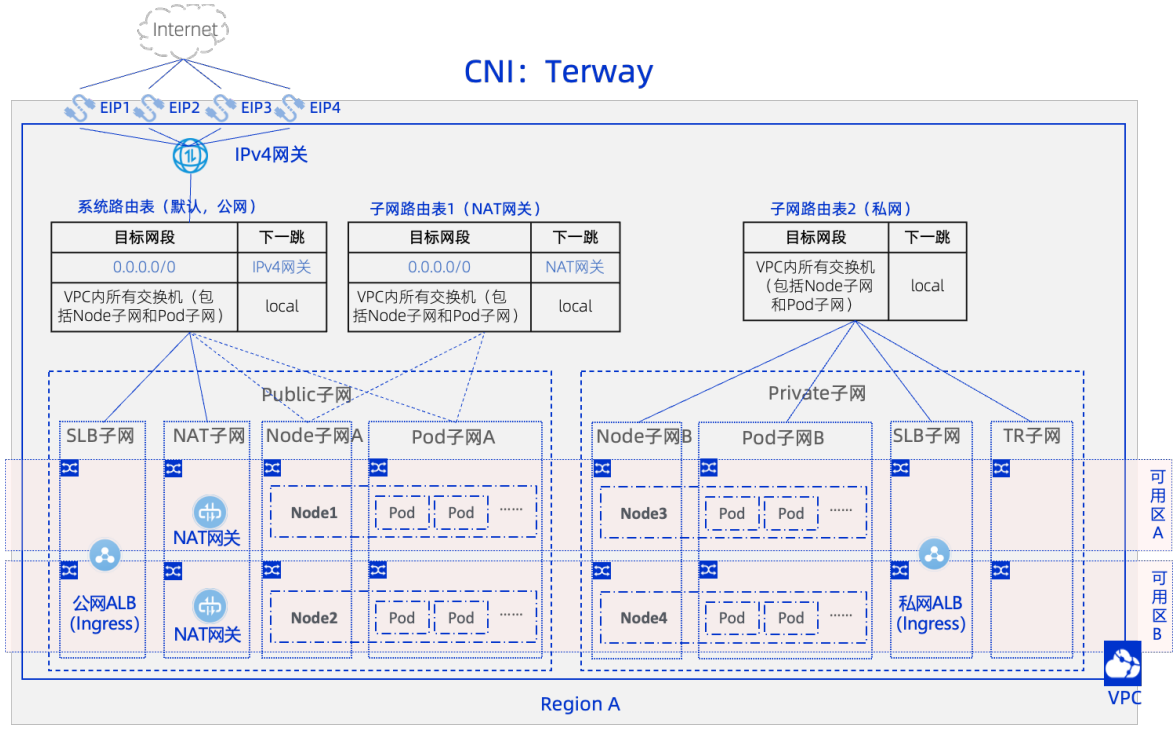
- **PAI-EAS 平台部署的 Agent 私网访问用户 VPC：**如上图橙色虚线箭头所示，PAI-EAS 实例需要先关联到用户 VPC（在用户 VPC 内插入 ENI 网卡），PAI-EAS 实例可以通过这个 ENI 网卡直接访问此用户 VPC，具体参见：通过网关进行公网或内网调用。如果要访问用户的其他 VPC，可以将其他 VPC 通过 CEN 与此 VPC 连通

(3) Function AI 平台 Paas 化部署模型的网络

后续补充。

(4) 容器 ACK 平台 Iaas 化部署模型的网络

当用户推理服务有极强的私有化部署诉求时，可以使用 ACK 专属集群部署推理模型。ACK 集群部署于用户 VPC 之中，所以需要规划和设计此 ACK 集群的 VPC 网络。下面简述一下设计原则，关于更细节的 ACK 网络说明可以参考：容器网络互联负载均衡服务发现。



确定地理位置

根据业务的服务对象所在位置选择地域（Region）。根据云资源储备情况、可用区时延情况选择可用区 AZ

确定集群规模

- 根据业务所需算力大小确定集群规模、根据业务峰谷弹性确定 POD 数量
- 根据前一页确定 CNI 使用 Terway 还是 Flannel

网段设计

- **VPC 自身网段**：根据 POD 数量评估地址个数、确定网段
- **VPC 交换机的网段（节点所在的子网）：**
    - 根据 Node 数量评估地址个数
    - 建议选择 2 个及机上可用区（高可靠）
- **Kubernetes Pod 的网段：**
    - 根据 Pod 数量评估地址个数
    - Pod 交换机需和节点交换机在同一可用区
    - Pod 网段使用 VPC 网段之内的地址（可与节点交换机网段重合）
    - Pod 网段不能与 Service 网段重叠
- **Kubernetes Service 的网段（ClusterIP 所在子网）：**
    - 集群内使用
    - 不能与 Node 及 Pod 的网段重叠
- **集群南北流量入口：**
    - ALB 做 Ingress 时，给 ALB 单独划分交换机和子网（2 个及以上可用区）

路由表设计

- **系统路由表（默认）：**

    (1) 创建 VPC 后，系统会默认创建一张系统路由表，此表有到 VPC 内所有交换机的路由

    (2) 当有出公网诉求并且配置 IPv4 网关时，在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 IPv4 网关

    (3) 此表被 Public 子网（公网 SLB 交换机、公网 NAT 交换机）关联

(4) 当 Node 子网内 Pod 绑定 EIP 想直出公网时，也需要关联此表

- **子网路由表 1（出公网）：**

(1) 当有出公网诉求并且配置 IPv4 网关时，如果 Pod 想要通过 NAT 网关出公网，则创建此表并关联

(2) 在此表中添加前缀为 0.0.0.0/0 路由下一跳指向 NAT 网关

- **子网路由表 2（私网）：**

(1) 当某些 Node 的 Pod 只需要内网通信、控制阻断公网能力时，配置 IPv4 网关

(2) 创建此表并 Node 交换机关联；3）此表中只有到 Node 子网的路由和响应 Pod 子网的路由

公网网关设计【可选&推荐】

- 当要控制服务器配置公网 IP 或绑定 EIP、公网 SLB 直出公网时，开启 IPv4 公网网关。公网网关开启后，所有交换机关联的路由表必须有到公网网关的路由才能出公网

安全防护设计【可选】

- 网络 ACL（NACL）可以控制交换机（子网）粒度流量阻断，网络 ACL 是无状态的
- 安全组可以控制 Node、Terway Pod 粒度的流量阻断，安全组是有状态的

网络监控和日志记录【可选】

- 配置 NIS 进行日常诊断观测，按需使用 flowlog 和流量镜像，并配置对应云监控告警

推理服务分发入口设计：

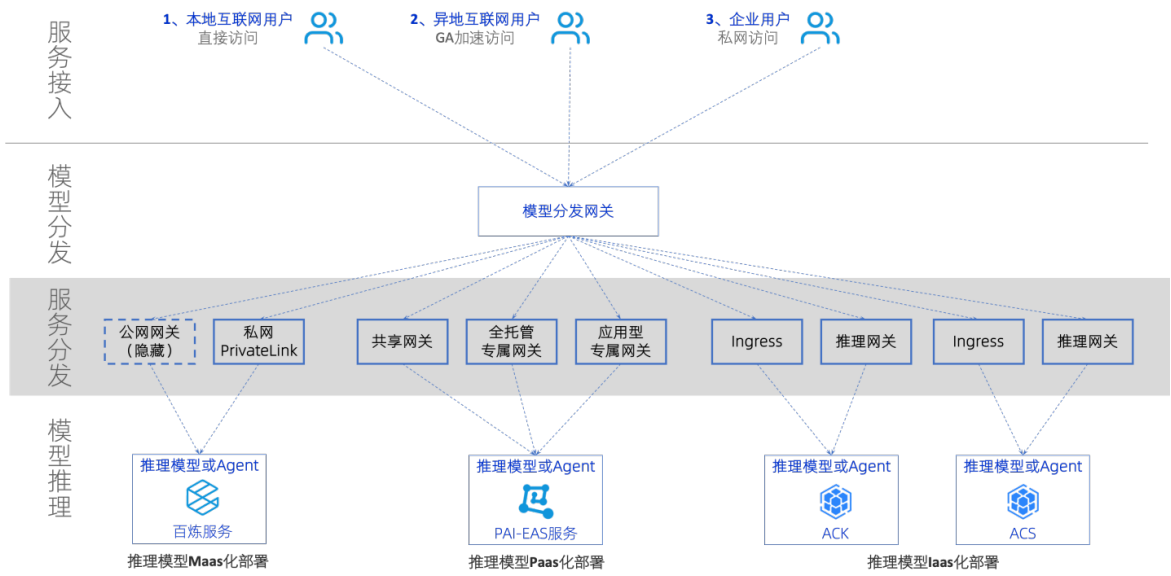
- 简单场景可以使用 Ingress 做 ACK 集群的推理服务入口
- 建议使用容器推理网关替代 Ingress 做服务分发网关，参见：生成式 AI 服务增强

(5) 容器 ACS 平台 Iaas 化部署模型的网络

当用户推理服务有极强的私有化部署诉求时，可以使用 ACS 部署推理模型。ACS 集群部署于用户 VPC 之中，所以需要规划和设计此 ACS 集群的 VPC 网络。网络的规划和设计原则可以参考前面的 ACK 章节（只是不需要考虑 Node 相关的部分），关于更细节的 ACS 网络说明可以参考：网络-容器计算服务(ACS)

3.6.2. 服务分发层的网络规划与设计

服务分发层部署各个模型自己的服务分发网关，将推理请求的服务转发到对应模型的端口上，同时可以实现限速、计量等功能。



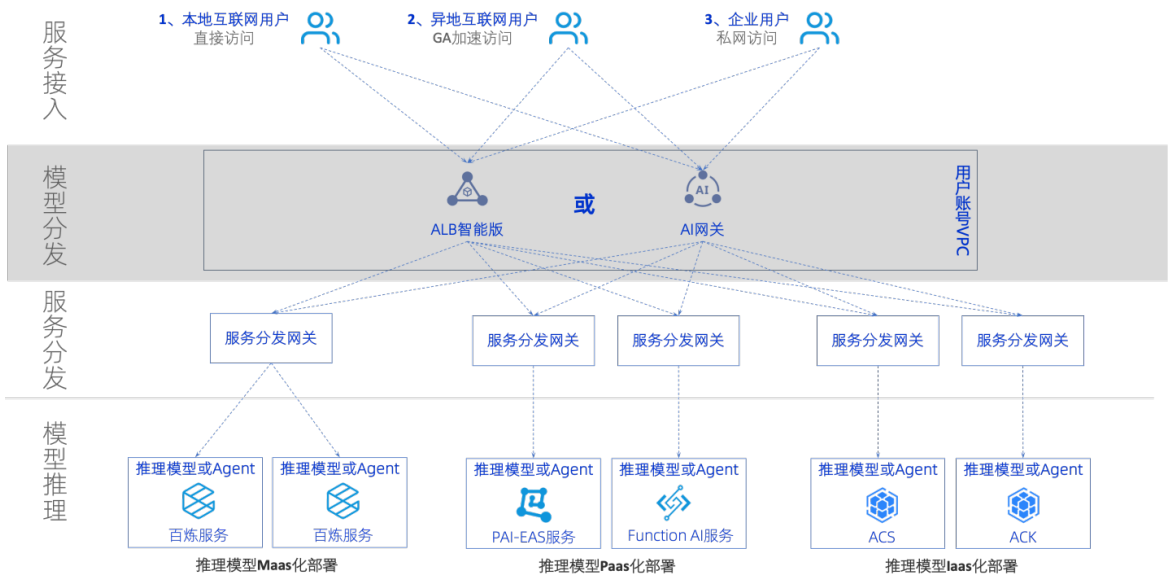
如上图所示，各个模型的服务分发网关：

- **百炼**：公网网关可以接入互联网的推理服务请求，此网关隐藏在百炼平台内、用户不可见；百炼私网接入可以直接使用私网域名调用 API，但需要使用 PrivateLink 打通百炼和用户 VPC 之间的私网通道
- **PAI-EAS**：共享网关、全托管专属网关、应用型专属网关都可以提供公网和私网推理服务请求接入。其中应用型专属网关生成在用户 VPC 内、用户 VPC 可以直接进行私网调用；共享网关、全托管专属网关托管在 PAI 平台内、需要将 PAI-EAS 实例关联到用户 VPC 后，用户 VPC 才可以进行私网调用
- **ACK 和 ACS**：简单推理场景可以使用常规的 Ingress 做推理服务请求接入，复杂推理场景建议使用推理网关做推理服务请求接入

以上服务分发网关的网络规划和设计在前面模型网络章节已有介绍，此处略去。

3.6.3. 模型分发层的网络规划与设计

当有模型 Fallback 场景、API Key 安全管理、成本控制等诉求时，通常在模型推理层部署不同类型的模型，此时需要规划模型分发层部署模型分发网关，它将不同的模型调用接口统一集成、根据调用的模型名称将推理请求分发到不同的后端模型上。

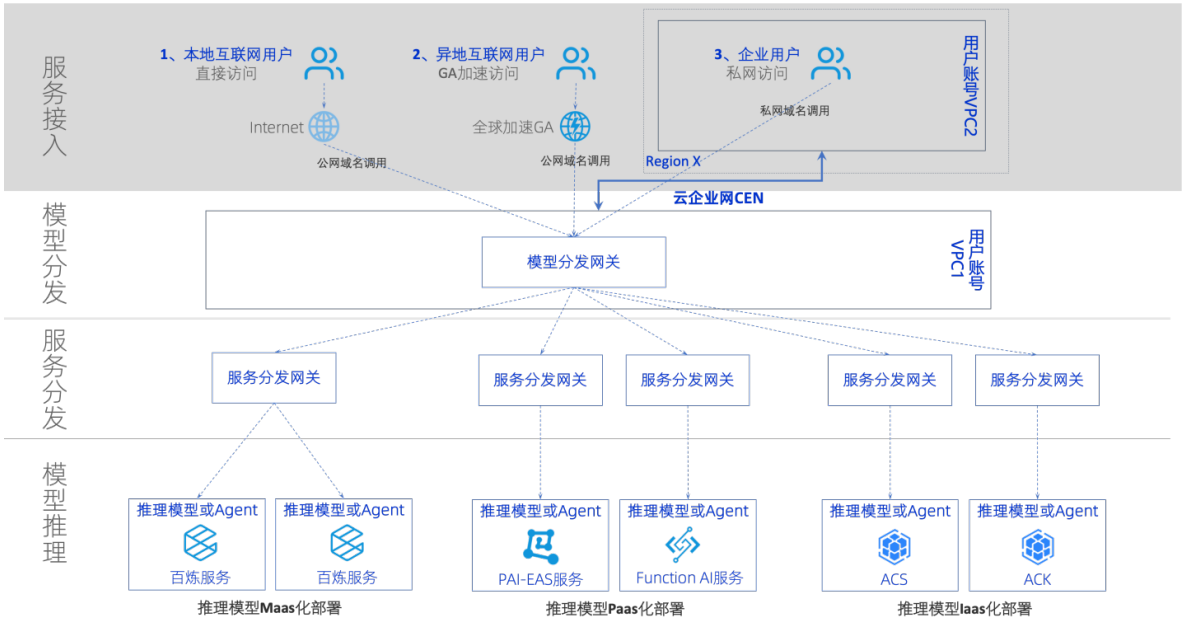


模型分发网关作为企业 AI 应用与模型服务、工具及其他 Agent 之间的核心连接组件，通过提供协议转换、安全防护、流量治理和统一观测等能力，协助企业构建和管理 AI 原生应用。模型分发网关可以使用阿里云 ALB 智能版或 AI 网关产品担任，它们都部署在用户 VPC 内，只需要简单的规划 VPC 及其交换机后即可部署：

- ALB 智能版发布中，后续补充部署指导文档
- AI 网关部署指导参见操作指南-API 网关(API Gateway)

3.6.4. 服务接入层的网络规划与设计

服务接入层，根据不同的终端访问规划和设计合适的网络产品来保证访问流畅和安全。



- **本地互联网用户访问推理服务：**直接使用模型分发网关的公网域名

- **异地互联网用户访问推理服务：部署全球加速 GA、加速模型分发网关的公网域名，参见：**配置全球加速实现加速访问指定域名的后端服务
- **阿里云上企业用户访问推理服务：**使用 CEN 打通 VPC2 和 VPC1 之间的私网路由，然后调用模型分发网关的私网域名。VPC 之间的 CEN 打通参见：通过云企业网实现同地域 VPC 互通

4. 身份权限

4.1. 概述

身份权限管理是 AI Landing Zone 安全体系的基石，它定义了“谁（Who）可以在何种条件下（Condition）对什么资源（What）进行何种操作（Action）”。本章旨在提供一个全面、精细且易于管理的身份权限框架，以确保您的 AI 平台、数据和模型资源在整个生命周期内都得到严格的保护，同时兼顾研发人员的敏捷性和效率。一个设计良好的身份权限体系，能够让正确的人与程序，在正确的时间，以正确的权限，安全地访问正确的资源。

4.2. 背景与挑战

AI 平台的身份权限管理不仅要遵循传统的 IT 安全原则，还面临着由其自身特性带来的新挑战：

- **参与角色多样化**：一个 AI 项目涉及数据工程师、算法科学家、应用开发者、运维人员等多种角色。每个角色都需要一套不同的、最小化的权限集合，传统的粗粒度权限模型难以满足这种精细化的授权需求。
- **资源类型复杂且敏感**：AI 平台管理的资源不仅包括云服务器、网络等基础设施，更涵盖了价值连城的数据资产（如标注数据集、用户隐私数据）和核心知识产权（如训练好的模型）。如何对这些新型资产进行精细化授权，是权限管理的核心难题。
- **多层次、非典型的权限模型**：许多 AI 平台（如 PAI、百炼）在标准的 IAM 之上，构建了自己独有的权限层级，如“工作空间权限”、“模型权限”等。这种多层次的权限体系增加了管理的复杂性，要求管理员必须同时理解并配置云平台和 AI 平台的两套权限。
- **程序化访问为主导**：与人工操作为主的传统系统不同，AI 平台中绝大多数操作是通过应用代码、自动化脚本发起的 API 调用。这使得对程序身份（如 API-Key、实例角色）的安全管理和凭证轮转变得至关重要，成为安全防护的焦点。

4.3. 具体方案

4.3.1. 方案示例

在上云规划过程中，企业需要识别不同身份对云资源的访问需求。典型的身份类型包括：

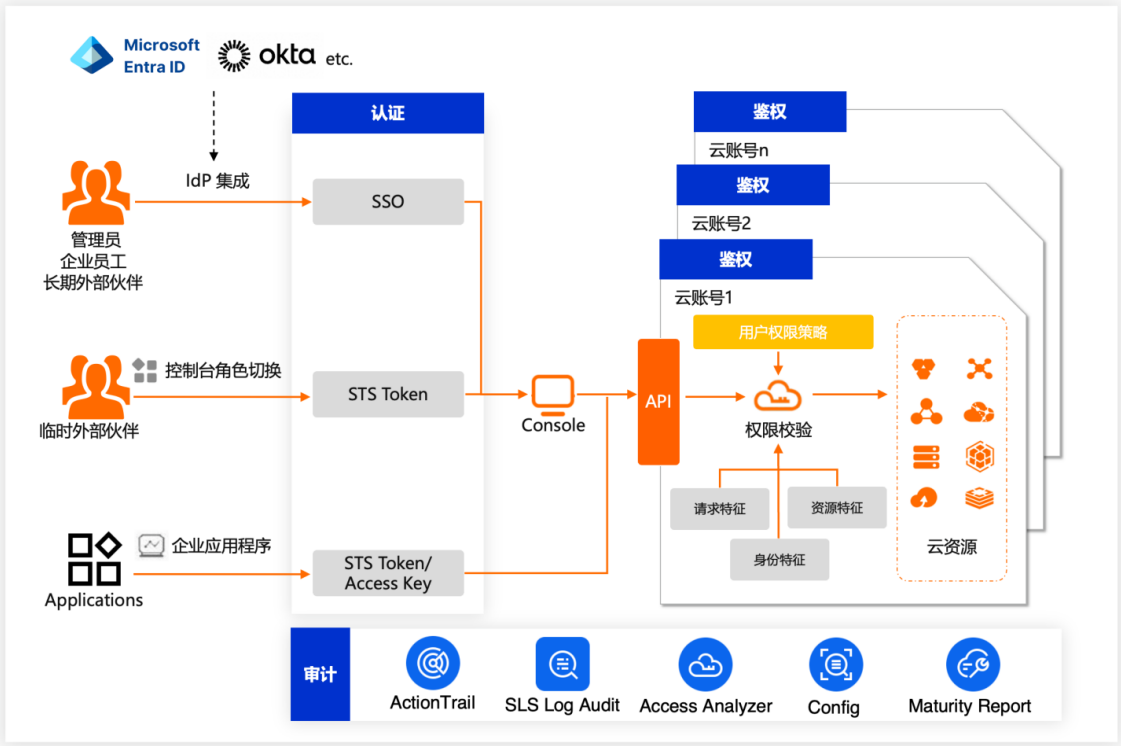
身份类型	权限需求
超级管理员	云管理团队，其成员需要拥有对云治理服务（如身份、权限、资源、合规、安全、网络、监控、备份等）的管理权限，无需拥有对计算、存储等业务所需资源的直接管理权限，但在需要时可以接管控制权。该团队还可以细分为财务管

	理员、安全合规管理员、网络管理员、数据库管理员等角色，侧重于云治理框架中某一方面的管理工作。
企业员工	各业务团队成员，他们需要使用归属于本部门的云资源进行开发、测试、运维等工作，一般不允许访问其他部门的资源，但如果出现跨部门合作，也应该可以被授权访问其他部门的资源。
企业外部人员	部分业务团队，需要合作伙伴获取本部门少量资源的读写权限。
企业客户	有些业务部门开发的应用提供代客户保存数据的服务，其业务场景需要允许客户直接访问由客户上传，但保存在企业的云存储中的数据。

为了满足上述需求，企业按照“最小够用”原则，对所有身份进行精细化权限管理：

- 对于超级管理员、企业员工使用 RAM 的"单点登录"（SingleSignOn,SSO）功能，将阿里云身份系统与企业自有身份系统打通，实现单点登录，并要求所有访问云资源的超级管理员、企业员工等使用 SSO 登录阿里云。
- 对于企业外部人员
    - 长期使用者，在企业身份系统中创建用户，采用同样的单点登录措施。
    - 临时使用者，在需要访问的云账号中创建 RAM 角色并授予带时间限制的有限资源访问权限，允许其使用自己持有的云账号进行角色切换登录。
- 对于企业客户，每次客户需要访问云资源时，由企业应用程序为其生成短时有效的安全访问令牌（STSToken），供客户在应用程序内使用。

身份和访问控制的整体架构示例如下：



该架构的核心优势包括：

- **统一身份认证：**打通企业内部身份和云端身份，员工离职可有效阻断访问，消除安全风险。
- **基于角色的访问控制：**基于角色的授权，避免权限管理混乱和权限过多，实现统一授权管理。
- **风险控制：**员工不持有用户密码，避免账户泄露风险。

4.3.2. 身份管理

身份管理是指对在云环境中执行操作的实体进行统一管理和认证。云上主要有两种身份类型：人员身份和程序身份。

(1) 身份管理的核心原则

在阿里云上，针对身份管理，基于安全的最小化原则，核心有两大原则：缩小暴露时长和缩小暴露面积。

- **缩小暴露时长：**尽可能使用临时身份或凭据替代固定身份或凭据，即使使用固定身份或凭据，也建议定期轮转。
- **缩小暴露面积：**尽可能安全保管密钥或凭据，避免在不同身份类型之间混用凭据或身份。

4.3.3. 人员身份管理

(1) 避免使用 Root 身份

在阿里云上，注册完一个阿里云账号后，即可通过用户名和密码的方式登录到阿里云控制台，登录成功后，即获得了 Root 身份。该身份具有该账号下所有的权限，一旦账号密码泄漏，风险极高。另外，如果

多人共用该身份，每个人都保有该账号的用户名和密码，会增加泄漏的可能。同时，多人共享的情况下，在云上的操作日志中无法区分出是组织中哪个人使用了该身份进行了操作，也无法进行进一步溯源。因此，除了极个别场景，应该尽可能的使用阿里云 RAM 身份进行云上资源的访问，避免使用阿里云账号的 Root 身份。

(2) 实现人员身份的统一认证

通过集中化的身份提供商（IdentityProvider，简称 IdP）来进行人员身份的统一认证，能够简化人员身份的管理，确保组织内在云上、云下的人员身份的一致性。当人员结构变更、人员入职、离职时，能够在一个地方完成人员身份的配置。对于云环境的使用者来说，也无需为其颁发额外的用户名和密码（如 RAM 用户名和密码），只需要保管好其在组织内的 IdP 中的身份和凭据即可。

阿里云支持基于 SAML2.0 协议的单点登录（SingleSignOn，简称 SSO）。在阿里云上，我们建议通过 RAMSSO 的方式跟组织内的 IdP 进行集成，实现人员身份的统一认证。对于云上有多个阿里云账号的复杂组织，还可以通过云 SSO 进行多账号下的 SSO 集中化配置，进一步实现多账号下的人员身份统一管理，提升管理效率。

AI 场景下的实践建议：

对于 AI 场景，建议优先使用 RAMRole 进行身份管理，因为角色身份更安全、便于权限管理。大多数 AI 服务（如百炼、PAI、ACK、OSS 等）都支持 RAMRole，建议采用云 SSO+RAMRole 的方式，通过云 SSO 创建访问配置模板，将企业 IdP 中的用户组映射到阿里云 RAM 角色，从而实现对 AI 服务访问人员的统一管理。

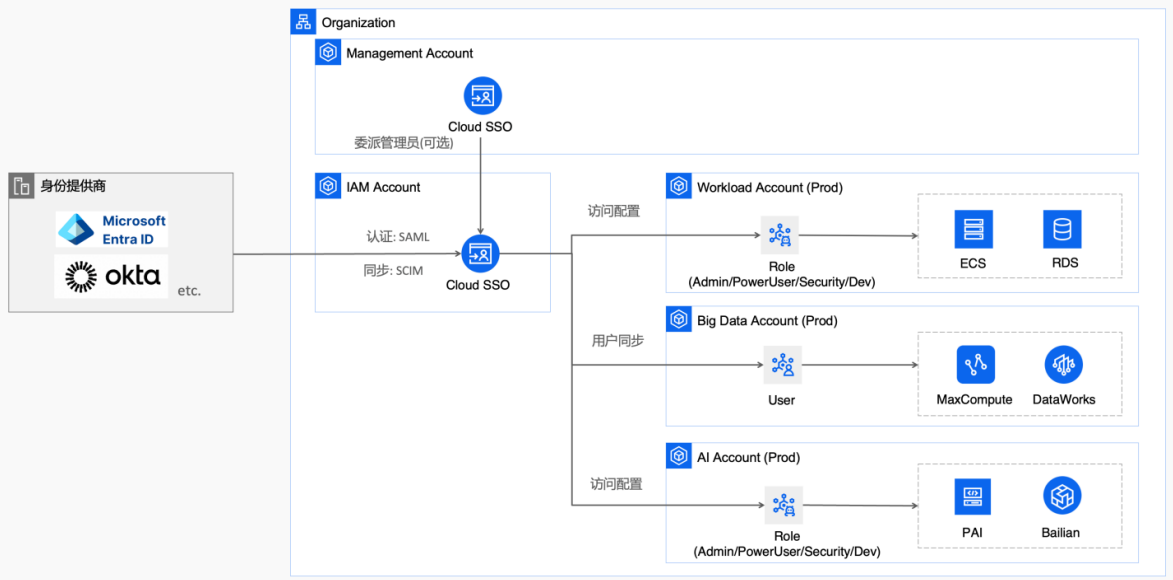
对于不支持 RAMRole 的 AI 服务，需要通过云 SSO 的用户同步功能，将用户直接同步到对应账号，采用用户 SSO 的方式进行访问。

典型的角色划分包括：

- **AI 模型管理员：**负责 AI 模型的全局管理，需要跨业务空间的管理权限。
- **AI 应用研发：**负责智能体开发、模型调用等应用研发工作，仅需要特定业务空间的访问权限。
- **数据管理员：**负责 AI 训练数据的准备和管理，需要数据相关的访问权限。

AI 场景下的 SSO 架构示例：

在 AILandingZone 场景下，建议采用集中化的 SSO 架构来统一管理多账号下的身份和访问。典型的架构示例如下：



该架构展示了 AI 场景下的 SSO 认证和访问流程：

1. **身份认证发起：**用户通过外部身份提供商（IdP），如 MicrosoftEntraID、Okta 等，发起登录请求。

2. **认证和同步：**IdP 通过 SAML2.0 协议将认证信息传递给 IAM 账号中的 CloudSSO，同时通过 SCIM 协议进行用户信息同步，确保云上用户身份与 IdP 中的身份保持一致。

3. **集中化身份管理：**IAM 账号作为核心身份管理中心，通过 CloudSSO 接收并处理来自 IdP 的认证请求。管理账号（可选）可以通过委派管理员的方式，对 CloudSSO 进行统一管控，实现身份管理的集中化控制。

4. **访问路由和权限分配：**根据用户身份和权限配置，CloudSSO 将用户路由到不同的生产账号，并根据账号类型采用不同的 SSO 方式：

   - **WorkloadAccount(Prod)：**采用角色 SSO 方式，通过访问配置模板，用户通过角色扮演（如 Admin、PowerUser、Security、Dev 等角色）访问该账号，管理 ECS、RDS 等工作负载资源。

   - **BigDataAccount(Prod)：**采用用户 SSO 方式，通过用户同步功能，直接将用户身份同步到该账号，用户可以直接访问 MaxCompute、DataWorks 等大数据服务，进行业务空间权限管理等操作。

   - **AIAccount(Prod)：**采用角色 SSO 方式，通过访问配置模板，用户通过角色扮演（如 Admin、PowerUser、Security、Dev 等角色）访问该账号，管理 PAI、Bailian 等 AI 基础设施和平台服务。

5. **资源访问：**用户完成 SSO 登录后，根据所分配的角色或用户权限，访问对应账号下的云服务和资源，进行相应的操作和管理。

架构设计要点

- **角色 SSOvs 用户 SSO：**
    - 角色 SSO：适用于支持 RAMRole 的服务（如百炼、ECS、PAI、ACK、GPU 等），通过角色扮演实现临时身份访问，更安全且便于权限管理。对于 AI 服务，应优先使用角色 SSO。
    - 用户 SSO：仅适用于不支持 RAMRole 的 AI 服务，需要直接的用户身份进行业务空间权限管理等操作。对于此类服务，必须使用用户 SSO 的方式。
- **访问配置模板化：**通过云 SSO 创建访问配置模板，可以实现权限策略的复用，针对不同账号应用相同的权限模板，降低管理复杂度。
- **多账号统一管理：**通过集中化的 IAM 账号管理 CloudSSO，实现所有生产账号的身份和权限统一管理，当人员变动时，只需在一个地方进行配置即可影响所有账号。

(3) 建立更安全的登录机制

对于人员身份的登录方式，往往是通过用户名和密码的方式进行。一旦用户名和密码泄漏，攻击者极有可能通过该身份登录阿里云，造成一些不可挽回的损失。因此，对于人员身份来说，保护好用户名和密码显得尤为重要。可以从以下几种方式提升登录方式的安全性：

- **提升密码强度：**如增加密码位数、混合使用数字、大小写字母、特殊字符等。针对阿里云 RAM 用户，管理员可以设置密码强度规则，强制 RAM 用户使用更复杂的密码，降低密码泄漏和被破解的风险。
- **避免密码混用：**在不同的服务、站点，或不同的用户共用一个密码，增加了密码的暴露面积，会增加密码泄漏的可能性。
- **定期轮转密码：**密码存在的时间越长，泄漏风险越高。通过定期重设密码，降低单个密码存在的时长，能够进一步降低密码泄漏风险。
- **使用多因素验证：**多因素认证 MFA（MultiFactorAuthentication）是一种简单有效的安全实践，在用户名和密码之外再增加一层安全保护，用于登录阿里云或进行敏感操作时的二次身份验证。建议为云上的人员身份都启用二次验证。

(4) 通过角色扮演代替固定身份

基于缩小暴露面积的原则，使用临时身份替代固定身份，能极大降低身份泄漏风险，同时对于人员身份来说，也能够抽象人员权限模型，如按人员职能进行角色划分，有利于规范化人员权限设置，提升管理效率。

在云上，建议通过单点登录（SingleSign-on，简称 SSO），基于角色扮演的方式，实现人员身份的管理。在 AI 场景下，通过云 SSO 创建访问配置，定义不同角色的权限模板，然后将用户或用户组映射到相应的访问配置，实现基于角色的统一管理。

4.3.4. 程序身份管理

在 AI 场景下，应用程序通常需要通过 API 调用 AI 服务，需要使用程序身份进行认证和授权。

(1) 不使用云账号 AccessKey

云账号 AccessKey 等同于阿里云账号的 Root 权限，也就是该账号内的完全管理权限，而且无法进行条件限制（例如：访问来源 IP 地址、访问时间等），也无法缩小权限，一旦泄漏风险极大。对于程序访问的场景，请使用 RAM 用户的 AccessKey 来进行阿里云 API 的调用。

(2) 避免共用 AccessKey

多个程序身份共用 AccessKey，或程序身份、人员身份共用 AccessKey 的情况下，AccessKey 所关联的权限需要包含所有身份的使用场景，导致权限扩大。另外，共用场景下，一处泄漏会导致所有应用都受到影响，风险扩大，同时增加了泄漏后的止血难度。

因此，针对不同应用、大型应用的不同模块、同一个应用（或模块）的不同环境（如生产环境、测试环境等），都建议创建不同的 AccessKey 供程序身份使用，每个 AccessKey 只有该场景下所需要的权限，避免共用 AccessKey 的情况。

(3) 定期轮转 AccessKey

同人员身份的用户名密码一样，AccessKey 创建和使用时间越长，泄漏的风险越高。每隔一段时间，通过创建新的 AccessKey，替换掉应用在使用的 AccessKey，并将旧 AccessKey 进行禁用和删除，实现 AccessKey 的定期轮转。另外，也可以通过阿里云密钥管理服务（KeyManagementService，简称 KMS）的凭据管家功能，实现自动化的定期 AccessKey 轮转。

(4) AI 场景下的 API-Key 管理

对于 AI 服务（如百炼），通常需要创建 API-Key 供应用程序调用。建议创建一个给应用程序用的 RAMUser（禁掉控制台登录），给这个 RAMUser 分配一个 API-Key。这样就不会出现因为用户离职之后导致的 Key 失效问题，同时也能实现程序身份与人员身份的隔离管理。

(5) 使用临时凭据代替固定凭据

通过给 RAM 用户或云账号的 Root 身份创建 AccessKey 供程序调用，都属于固定凭据类型。一旦创建出来，在删除之前，就是由固定的 AccessKeyID 和 AccessKeySecret 组成了该凭据。使用固定凭据会造成很多风险，比如应用研发人员将固定 AccessKey 写入了代码中，并将其上传到了 Github 等公开仓库，造成了 AccessKey 泄漏，最终导致业务受损。在阿里云上，我们建议尽可能通过角色扮演的方式获取临时凭据 STSToken，代替固定 AccessKey 的使用。

针对不同类型的云上应用部署方式，阿里云提供了相应的功能，集成了 STSToken 凭据的使用：

- **ECS 实例：**通过实例 RAM 角色，将 RAM 角色跟实例进行绑定，应用程序中即可通过实例元数据服务获取临时授权 Token。

- **容器服务（ACK）：**通过 RRSA 功能，实现 RAM 角色和指定 ServiceAccount 进行绑定，在 Pod 维度即可扮演对应角色实现 STSToken 的获取。
- **函数计算（FC）：**可以为对应函数所属服务授予函数计算访问其他云服务的权限，实现 STSToken 的获取。

4.3.5. Agent 身份管理

AIAgent 是具备规划、推理和代表人类或其他系统采取自主行动能力的实体，这类身份需要单独管理。

(1) 统一管理所有 Agent 身份

AI Agent 是新的攻击面，需要对企业所有的 AI Agent 进行统一的管理，自动发现并注册每个 AI Agent 的身份，避免未纳入管理、监控的 AI Agent 发生非预期的安全事件。

建议通过 Agent Identity 统一管理各类 AI Agent，如百炼平台的 Agent、Dify 平台的 Agent，以及基于任何 Agent 框架构建的 AI Agent。并基于 Agent Identity 持续监控、管理用户及程序在 AI Agent 的访问权限及使用情况。

(2) 基于组织 IdP 对 Agent 用户进行认证

IdP 身份提供商集中管理企业员工身份或者用户身份，基于 IdP 建立 Agent 的信任关系并提供对 Agent 的访问能力，可以确保每次用户对 Agent 的访问，都是可信、可验证的，同时当人员变动，如新加入、离开时，对 Agent 的访问权限也随之赋予、撤销。

建议通过 Agent Identity 配置 Agent 对 IdP 的信任关系，从而实现基于 IdP 的无密钥统一认证。

(3) 统一管理 Agent 需要使用的凭据

AI Agent 在访问大模型服务(如阿里云百炼)、阿里云服务(如 OSS)、企业服务(如 CRM)、SaaS 服务(如钉钉)时，需要使用各类访问凭据，需要对凭证进行统一的管理。

建议通过 Agent Identity 的凭证提供商进行统一管理，基于 KMS 对凭证进行安全存储，并实现：只有正确的人，使用正确的 AI Agent，才能临时获取到对应服务的范文凭证。

4.3.6. 访问控制

访问控制是指控制某个身份在什么条件下对哪些资源能够执行哪些操作。阿里云上的授权方式分为以下几种：

- **基于身份的授权：**主要是指针对 RAM 用户、用户组或角色进行授权。
- **基于资源的授权：**部分云产品支持为某个特定资源绑定权限。如 OSS 的 BucketPolicy。
- **管控策略（ControlPolicy）：**针对启用了资源目录的多账号组织，基于资源结构（资源夹或成员）的访问控制策略，可以统一管理资源目录各层级内资源访问的权限边界。

- **会话策略（SessionPolicy）：**在角色扮演的过程中，可以指定一个会话策略，定义本次扮演中可以获得权限，进一步缩小角色权限的范围。

无论基于何种授权方式，合理的权限设置能够阻止未经授权的访问，保护云上资产和数据的安全。因此，云上的权限管理的核心原则就是**权限最小化**，只给身份授予必要的权限，确保权限最小够用。

(1) 人员身份的权限管理

基于人员职能进行授权

对于组织来说，不同职责的人需要访问云上不同类型的资源。对于 AI 场景，建议针对以下角色进行权限设计：

- **全局云管理员：**拥有企业在阿里云上资源的全部权限。
- **AI 模型管理员：**拥有 AI 服务（如百炼）的全局管理权限，可以跨业务空间进行管理，负责模型的分配和业务空间的规划。
- **AI 应用研发：**拥有特定业务空间的访问权限，可以进行智能体开发、模型调用等工作。
- **AI 数据管理员：**拥有数据相关的访问权限，负责训练数据的准备和管理。
- **网络管理员：**拥有对各类网络产品的管理权限。
- **安全管理员：**拥有全部安全产品的管理权限。
- **合规管理员：**拥有合规相关产品的管理权限。

在对职能权限进行抽象后，可以通过将人员身份加入到指定职能用户组的方式进行组织，提升授权效率。

按资源范围授权

虽然权限管理的核心原则是最小化授权，但如果为组织中的每个人员身份都定制化权限，对于大型组织来说，会大大降低管理效率。因此，按照人员所管理的业务应用对应的资源范围进行授权，能够简化授权逻辑，提高权限策略复用率，进而在权限最小化和管理效率中取得平衡。

在云上，建议通过阿里云账号或资源组两种方式，区分不同业务应用的资源。如果企业使用多个阿里云账号管理云资源，则可以使用资源目录构建企业组织结构，对账号和资源进行集中、有序地管理，不同的业务应用按账号维度进行区分，授权时应用范围选择整个云账号。如果企业使用一个阿里云账号管理所有云资源，各个项目组可以使用资源组作为资源隔离和权限管理的容器，在授权时应用范围选择指定的资源组。

AI 场景下的实践建议：

对于 AI 场景，特别是使用百炼等 MaaS 平台时，建议通过业务空间进行资源隔离和权限管理：

- **业务空间规划：**模型管理员负责规划业务空间，按照不同的业务应用、项目组或部门创建不同的业务空间。
- **业务空间权限：**通过 RAMRole（优先）或 RAMUser（对于不支持 RAMRole 的服务）的方式将用户或用户组授权访问特定业务空间，实现按业务空间进行权限隔离。
- **跨业务空间管理权限：**全局管理权限（如 AliyunBailianFullAccess）允许用户进行跨业务空间的管理操作（如创建业务空间、删除业务空间、分配模型权限等），但不包含对具体业务空间内资源的访问权限（如调用模型、创建应用等）。业务空间内的资源访问权限需要单独授予。因此，全局管理权限建议只分配给模型管理员，用于统一管理业务空间和模型分配。

AI 服务的多层次权限管理

AI 服务（如百炼）通常具有多层次的权限体系，需要分别进行精细化管控：

- **RAM 管控面权限：**通过 RAMRole（优先）或 RAMUser（对于不支持 RAMRole 的服务）授予用户在 AI 服务上的基础访问权限，如 AliyunBailianFullAccess（全局管理）、AliyunBailianReadOnlyAccess（只读）、AliyunBailianDataFullAccess（数据管理）等。
- **业务空间权限：**控制用户能否访问特定的业务空间。这是 AI 场景下资源隔离的核心机制。
- **模型权限：**控制业务空间能否调用、训练和部署某个模型。模型权限与用户权限分离，由模型管理员统一分配给不同业务空间，与空间成员的权限无关。

实践建议：

- **模型权限统一管理**：建议由模型管理员统一为不同业务空间分配模型权限及相关限流配置，避免权限分散管理。
- **权限分离原则：**理解 RAM 权限、业务空间权限、模型权限之间的关系，建立清晰的权限管理体系。

(2) 程序身份的权限管理

精细化授权

除一些特定业务场景外，应用程序所需要访问的阿里云资源，对应进行的操作是可以预期，尽可能的通过自定义权限策略来定义该程序身份所需要的最小权限。对于 AI 场景，应用程序通常只需要调用特定模型权限，不应该授予全局管理权限。

相反，如果直接使用系统策略，给该程序授予 AliyunBailianFullAccess 权限，那么一旦该程序身份泄漏，攻击者就有该云账号下所有 AI 服务的所有权限，风险极高。

AI 场景下的程序身份权限设计

对于 AI 应用，建议根据应用的实际需求进行精细化授权：

- **模型调用场景：**仅授予特定模型的调用权限，限制调用频率和资源配额。
- **数据访问场景：**仅授予特定数据源的读取权限，如特定 OSSBucket 的读取权限。
- **训练场景：**仅授予特定模型的训练权限，限制训练资源的使用。

(3) Agent 身份的权限管理

明确可使用 Agent 的用户范围

AI Agent 的调用方以及 AI Agent 自身，都需要控制允许使用 Agent 的用户范围。需要在 AI Agent 上配置允许的用户域(如企业内部的员工，或者经过某 IdP 认证的用户)，从而实现对 Agent 访问的纵深防御。

建议使用 Agent Identity 的身份提供商，并配置允许的 IdP 或者允许的受众，从而避免未经授权的 Agent 访问。

Agent 使用委托代表用户访问资源

对于公共资源，AI Agent 可能以 Agent 自己的身份权限进行资源访问，但对于企业内、用户私有数据，应该使用委托授权的方式，经用户授权后对其访问。

建议使用 Agent Identity 的 OAuth2 凭证提供商，全托管对下游服务、工具的访问授权，在 AI Agent 需要访问服务、数据时，发起 OAuth 授权流程，即时获取对服务、工具的最小访问权限。

对 Agent 获取凭据进行细粒度权限控制

AI Agent 需要访问各类服务、工具，访问这些服务、工具时都需要相应的凭证（如 API Key、Token），AI Agent 在获取凭证时应该遵循零信任理念，对运行时的上下文进行充分验证，通过实时细粒度的权限控制，确保只有正确人在使用正确的 AI Agent 才能获取到凭证。

建议使用 Agent Identity 的凭证提供商能力，在全托管各类访问凭证后，对 AI Agent 在什么用户的什么情况下才能获取哪个凭证提供商的凭证进行细粒度的权限控制，最大程度降低凭证暴露、泄露的风险。

(4) 通用的最佳实践

定期审查权限

在授权完成后，还需要定期对权限的授予进行审计确保每个身份的权限持续满足最小够用原则。需要重点关注的场景：

- **特权身份：**比如拥有所有产品权限的管理员，拥有 RAM 等管理与治理相关产品所有权限的身份，都属于重点审计对象。确保这些身份拥有这些特权是合理的。
- **闲置权限：**除了特权身份外，对于其他产品权限，也可以结合云上的操作审计日志，判断该身份的权限是否有闲置情况。

- **AI 场景特殊审查：**定期审查模型权限分配是否合理，API-Key 使用情况等。

设置权限边界

对于云上有多个阿里云账号的组织，可以基于资源目录管控策略，限制成员账号内的 RAM 身份权限范围，禁用一些高危操作降低身份泄露风险。如禁止成员删除域名或修改域名解析记录、禁止成员删除日志记录等。

AI 场景下的权限边界设置：

- 限制成员账号对 AI 服务的全局配置修改权限。
- 限制成员账号删除业务空间、删除模型等危险操作。
- 限制 API-Key 的创建和使用范围。

在绑定管控策略前，建议先进行局部小范围测试，确保策略的有效性与预期一致，然后再绑定到全部目标节点（资源夹、成员）。

人员变动时的权限管理

当企业发生访问云资源的人员配置（新增用户，删除用户，变更用户权限）时，需要进行相应的权限调整：

(1) 新增用户：应在 IdP 中将其加入到已经配置了 SSO 访问的用户组，并根据其职能授予相应的访问配置。

(2) 删除用户：可以直接删除 IdP 中的用户，IdP 通常会自动移除其访问权限。对于 AI 场景，还需要注意检查该用户是否有创建的 API-Key，及时进行清理。

(3) 用户权限发生修改：应修改其用户组配置或在云 SSO 中调整其访问配置。

(4) 转岗场景：需要及时调整用户在不同业务空间的访问权限，确保权限与职责一致。

多账号场景下的权限管理

对于使用多账号架构的企业，建议：

- 在管理账号开通云 SSO，统一管理所有成员账号的身份和权限。
- 通过访问配置模板，实现权限策略的复用，降低管理复杂度。
- 利用资源目录的管控策略，统一设置权限边界，确保安全合规。

4.4. 总结

为 AI Landing Zone 构建一个强大而灵活的身份权限体系，是保障其长期安全、稳定运行的关键。一个成熟的体系应在便利性、安全性和可管理性之间取得平衡。

核心建议是：

- **以统一身份源为核心：**通过云 SSO 与企业自有身份提供商（IdP）集成，实现单点登录和集中化的身份生命周期管理，这是提升管理效率和安全性的基石。
- **贯彻最小权限原则：**综合运用 RAM 策略、资源组、业务空间以及特定于 AI 服务的权限模型，确保每个身份（无论是人还是程序）仅拥有其完成任务所必需的最少权限。
- **优先使用临时凭证：**大力推广基于 RAM 角色的访问方式，为人员提供角色 SSO，为应用程序提供实例 RAM 角色或 RRSA，最大限度地减少静态 AccessKey 的使用，降低凭证泄露风险。
- **分层管理 AI 权限：**清晰地分离和管理 RAM 管控面权限、业务空间权限和模型权限，建立与 AI 平台特性相匹配的、权责分明的多层次授权体系。
- **持续审计与治理：**定期审查权限配置，利用访问分析器等工具发现闲置或过大的权限，并通过管控策略设置不可逾越的安全“护栏”，确保持续合规。

遵循这些原则，您可以构建一个既能满足严格安全合规要求，又能支持 AI 业务敏捷迭代的身份权限管理平台。

5. 安全防护

5.1. 概述

在构建 AI Landing Zone (AI LZ) 的过程中，确保环境的稳定与安全是所有创新的基石。本文旨在系统性地梳理 AI 技术在基础设施、模型及应用层面面临的核心安全挑战，并结合业界领先的安全框架与阿里云的最佳实践，提供一套端到端、全生命周期的安全防护方案。我们的目标是通过可信、可控的技术手段，保障您的 AI 业务在高速发展的同时，构筑坚实的安全防线。

5.2. 背景与挑战

随着 AI 技术的持续火热，安全事件频发，保护 AI 资产已成为企业发展的关键。从根本上说，AI 安全挑战贯穿了从基础设施到模型再到应用的每一个环节，具体可分为以下三个维度：

5.2.1. AI 基础设施风险

AI 的运行依赖于稳固的基础设施，但这一层也面临着严峻的传统与新型安全风险的叠加挑战。

- **系统与数据层面**：攻击者不仅会利用系统漏洞植入后门，导致凭证和数据泄露；还会滥用宝贵的算力资源进行挖矿等非法活动。更隐蔽的威胁是“数据投毒”，通过在训练阶段注入恶意样本，从源头上破坏模型的可靠性，甚至植入“后门”。
- **网络与身份层面**：网络隔离失效可能导致攻击者在内网横向移动，造成更大范围的破坏。同时，API 接口的滥用、访问令牌的泄露以及内部人员的越权访问，都可能为数据窃取和系统瘫痪打开方便之门。

5.2.2. AI 模型风险

作为 AI 系统的“大脑”，模型本身已成为新型攻击的重灾区，其风险主要集中在合规与数据安全两个方面。

- **合规治理挑战**：攻击者可以构造恶意内容（如包含恶意代码的文件）作为输入，对 AI 系统本身进行攻击。同时，通过“越狱提示词”等手段，可以诱导模型生成色情、暴力等非法或不道德的违规内容。此外，模型自身产生的“幻觉”现象，也可能引发法律和伦理风险。
- **模型数据安全**：模型在交互过程中可能会无意间泄露其训练数据中的敏感信息。攻击者可以通过精心设计的提示词，如“奶奶漏洞”，从模型中“套取”如正版软件序列号、个人隐私等关键信息。

5.2.3. AI 应用风险

在应用层面，传统的 Web 安全风险依然存在，并与 AI 场景下的新威胁相结合，形成了更复杂的攻击面。

- **资源消耗型攻击**：攻击者可通过构造大量消耗计算资源的请求与大模型交互，进行 DDoS 攻击，不仅影响正常用户服务，还会带来高昂的计算成本。

- **传统 Web 攻击**：针对 AI 服务的 Web 界面，SQL 注入、XSS 等传统网络攻击手段依然有效。
- **智能化内容爬取**：新型的“AI 爬虫”能更智能地模拟人类行为，大规模抓取受版权保护的内容用于训练其他模型或商业用途，给内容原创者带来巨大损失。

5.3. 具体方案

5.3.1. AI 基础设施与应用安全方案

围绕系统安全、数据安全、网络安全、身份安全、应用安全的风险，需要建立纵深防护体系，涵盖从操作系统到应用的整个技术栈。

(1) 系统安全

系统安全防护的核心是围绕资产、漏洞、安全配置建立全方位的检测和监控机制，重点就是做好 AI 资产的识别和 AI 漏洞的检测，以便能够形成完善的安全态势感知。

- **基线检查：**使用 CIS 基准、云平台最佳实践等安全配置标准对系统进行检查，利用自动化工具（AI-SPM）进行逐项匹配。
- **配置优化：**移除或禁用容器镜像、虚拟机及宿主机操作系统上不必要的服务、端口和用户。
- **漏洞扫描：**定期对主机、容器、Serverless 等环境进行漏洞扫描，及时发现和修复已知漏洞。
- **资产管理：**建立并维护 AI 物料清单（AI-BOM），清晰掌握所有 AI 组件的来源和依赖关系，以便在出现漏洞时快速定位影响范围。
- **运行时保护：**在主机和网络层部署监控系统，检测异常行为（如异常进程、可疑网络连接、密码爆破等）。部署容器运行时的安全工具，监控容器行为，防止容器逃逸等攻击。
- **镜像扫描：**在 CI/CD 流水线中集成自动化安全扫描（SAST、SCA、容器扫描），只有通过安全检查的代码和镜像才能进入下一阶段。
- **可信任源：**建立内部授信的模型仓库，对引入的第三方预训练模型进行严格的安全评估和扫描。
- **数据安全**
- 数据安全防护的核心是敏感数据识别和处理，通过加密、脱敏以及异常行为审计等方式，对数据库、存储、大数据中的敏感数据进行识别，联动应用的读写行为，监控敏感数据的流转，避免泄露。
- **传输加密：**确保数据在网络传输过程中（如节点间通信）使用 TLS 等加密协议。
- **静态加密：**对训练数据、模型文件等静态数据进行加密存储。
- **数据识别：**对 AILZ 涉及到的数据库、存储、大数据等组件中存储的数据进行敏感数据识别，按照不同敏感等级进行标注，便于后续的数据处理

- **数据脱敏：**对于部分训练中不需要使用到完整敏感数据的情况，可通过脱敏算法进行特殊处理，避免完整信息被直接使用（如手机号，对中间四位数字加*）
- **数据审计：**针对敏感数据的异常读写行为进行审计和告警

(2) 网络安全

网络安全防护的核心是隔离和访问控制，通过黑、白名单机制，严格控制内网和公网的流量、内网横向的互访流量，避免未授权的访问流量，也减少入侵后的风险扩散半径。

- **区域隔离：**将 AI 基础设施划分为独立的安全域（如训练集群、推理服务区、数据存储区），通过防火墙、网络 ACL、主机安全组的网络安全策略严格控制域间通信，遵循最小权限原则。
- **公网管控：**针对训练集群需要访问公网的场景做严格的白名单限制（如仅开发对 **Github.com、Aliyun.com 的主动访问**）

(3) 身份安全

身份安全防护的核心是认证和权限管控，通过统一登录/认证、MFA 等原则和技术手段，确保每个用户在其自身职责范围内仅拥有最小权限，避免越权访问和凭据泄露后的风险扩大。

- **统一认证：**使用 IAM、ActiveDirectory 等统一认证系统进行统一的身份管理和生命周期管理。
- **权限隔离：**为用户、运维、财务等账户分配完成其任务所必需的最小权限。例如，数据科学家不需要生产环境的写入权限。
- **操作审计：**通过堡垒机对需要访问的资源进行统一纳管，实现权限管控和操作审计。
- **凭据管理：**将所有 API 密钥、密码、证书等敏感信息存储在专业的密钥管理工具（如 AliyunKMS、AWSSecretsManager）中，杜绝在代码或配置文件中硬编码
- **零信任：**通过 SASE 实现终端安全管理，对运维、数据分析、三方服务等不同身份的人员、设备进行管控，包括设备、账号、权限的一致性，终端设备的环境风险，终端上的数据外发管控等。

(4) 应用安全

应用安全防护的核心和传统 LZ 基本保持一致，通过 DDoS、WAF 等安全产品，保护 AI 应用暴露在公网上的 API 接口以及 Web 服务。

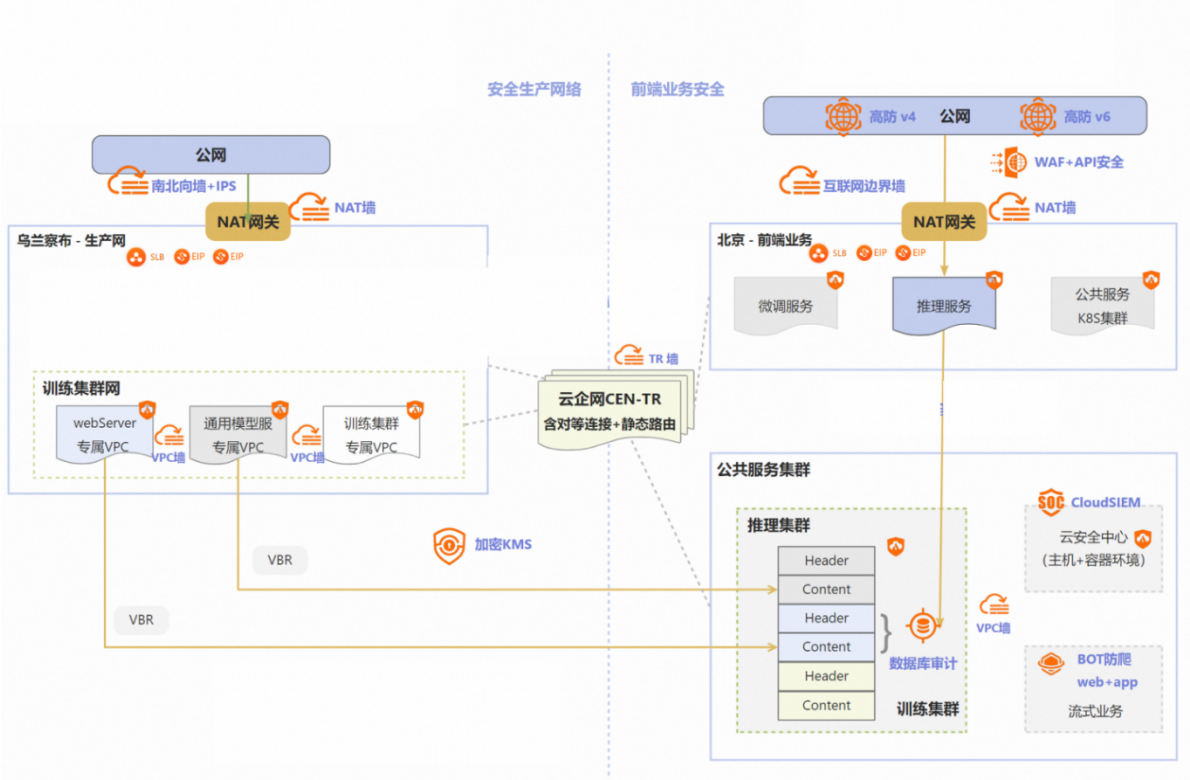
- 抗 DDoS：部署 DDoS 防护能力，针对流量型攻击、资源型攻击进行清洗
- Web 防护：标准的 Web 应用防护，针对 SQL 注入、XSS 等 Web 攻击进行拦截
- API 安全：识别异常流量（如来自单一 IP 的海量查询、异常的请求时序等）、API 风险检测（如未授权、弱口令、信息泄露等）

(5) 产品推荐

产品	能力	优势	覆盖风险
云安全中心	AI-BOM、AI-SPM、CWPP 等	从主机安全、容器安全延伸至 Serverless、PAI、灵骏智算等 AI 原生工作负载，支持 one-agent 与 agentless 双模式，实现对 AI 模型训练集群、推理服务及云原生资产的全生命周期防护，提供真正的算力平台统一防护能力。	系统安全
数据安全中心	敏感数据识别、UEBA	为云上数据源提供一体化的敏感数据识别、脱敏、加密、审计能力。支持常见的结构化数据、非结构化数据和大数据产品，例如对象存储 OSS、云数据库 RDS、MaxCompute 等。	数据安全
云防火墙	南北向、东西向访问控制	支持纳管所有公网资产、支持 CEN/CEN-TR 所有横向流量的访问控制，云上南北向、东西向流量一键接入防护。	网络安全
KMS	云产品落盘加密、凭据托管	针对云上 90+产品提供加密能力，覆盖模型数据准备、模型训练与微调、模型应用与部署三个阶段。通过自动轮转、一键开通等原生优势，提供便捷的凭据管理。	数据安全 身份安全
IDaaS	身份认证、统一登录、MFA	支持多云场景下的无 AK 方案，大模型应用可通过临时凭据访问云资源，根据场景动态调整访问权限，避免权限常驻，支持在整个链路中不使用永久凭证提升安全性。	身份安全

堡垒机	运维审计、权限管控	为 AI 资产提供便捷云上运维体验，轻量化部署、多 VPC 一键连通运维，云上资产深度集成，资产、凭据自动同步，特权账号智能识别。	身份安全
SASE	DLP、终端准入、零信任	将安全能力下沉至终端（PC、手机等），为多分支或门店、远程和移动办公场景的企业客户，提供即开即用的远程零信任访问、内网访问行为审计、办公数据保护、办公网准入等能力	身份安全
WAF	Web 应用防护、API 安全	针对 AI 业务敏感数据流转实现全线监控，不仅能秒级发现大模型相关的 API 资产，也能自动化探测隐私数据泄露风险，满足合规要求	应用安全
DDoS	DDoS 攻击清洗	部署简便、BGP 网络质量高时延小、防护能力大、系统稳定可用、防护精准，以及先进的 AI 智能防护技术	应用安全

最佳实践架构图



5.3.2. AI 模型安全防护方案

AI 模型防护实际由两部分组成，一部分是模型内生安全能力，即模型本身在训练、微调过程中对自身输入、输出的不断完善；另一部分是模型外挂安全能力，阿里云称为 AI 安全护栏。模型内生安全能力由模型供应商负责，我们重点讨论 AI 安全护栏的相关方案。

(1) 合规治理

- **提示词攻击检测：**针对生成式 AI 的注入式攻击，精准识别越狱指令、角色扮演诱导、系统指令篡改等对抗性攻击行为。
- **内容合规检测：**对生成式 AI 输入输出的文本内容进行多维度合规审查，覆盖涉政敏感、色情低俗、偏见歧视、不良价值观等风险类别，确保 AI 生成内容符合法律法规与平台规范。

(2) 数据安全

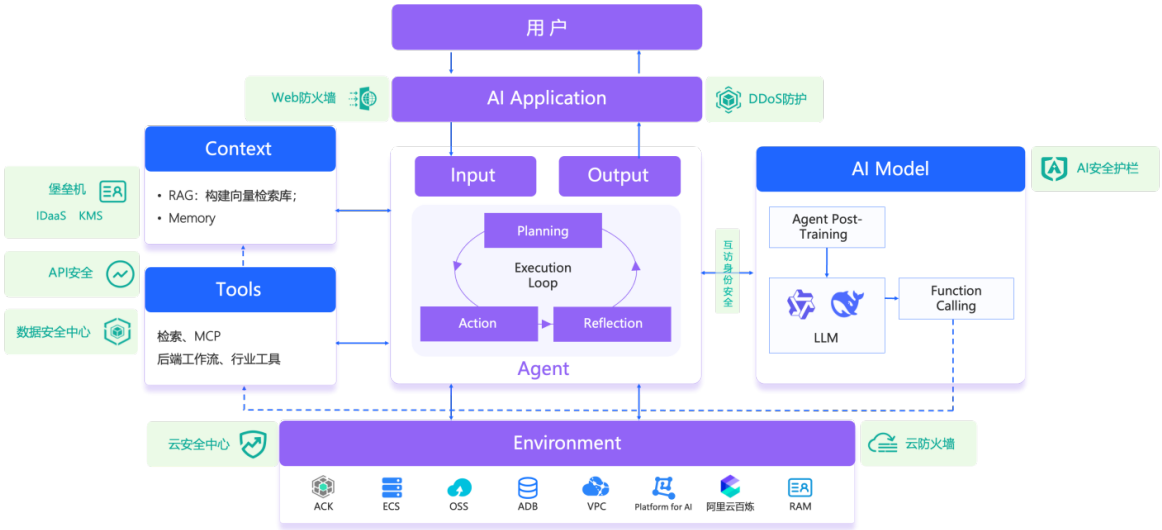
- **敏感内容检测：**深度检测 AI 交互过程中可能泄露的隐私数据与敏感信息，支持涉及个人隐私、企业隐私等敏感内容的识别，防范训练数据泄露与对话信息外溢风险。
- **可信计算：**基于硬件级 TEE 的安全计算环境（如 IntelTDX），确保数据在传输和计算过程中始终处于加密隔离的“飞地”中，防止未授权访问或篡改。通过双向远程证明技术（如 DCAP），业务后端可实时验证 MaaS 的 TEE 环境完整性，包括硬件平台、操作系统及应用的预期状态，确保计算环境未被篡改。

(3) 产品推荐

围绕 AI 模型的风险管控核心就是对输入、输出的检测能力，AI 安全护栏从提示词攻击、敏感内容检测、内容合规检测三个方面，完整覆盖了 AI 输入、输出的所有风险场景，一站式解决 AI 模型的防护方案。

5.4. 总结

构建安全可信的 AI Landing Zone 是一项超越单点防御的系统性工程。面对从基础设施到上层应用、从传统漏洞到新型 AI 攻击的复合性挑战，我们必须采取“纵深防御”与“原生安全”相结合的策略，构建覆盖 AI 全链路的分层防护体系。这套体系不仅旨在抵御当前威胁，更致力于打造具备韧性与弹性的安全架构，为 AI 技术的长期、健康与可持续发展保驾护航。



6. 合规审计

6.1. 概述

在 AI 时代，合规审计不再是传统的 IT 检查，而是保障企业 AI 战略可持续发展的核心支柱。它旨在通过系统性的检测、治理和追溯机制，确保 AI 系统在数据安全、模型伦理、内容合规等全生命周期内，始终满足内外部法规要求和风险控制标准。本章将提供一套整合了 AI 安全产品与治理能力的合规审计方案，帮助您构建一个安全、可信、负责任的 AI Landing Zone。

6.2. 背景与挑战

AI 技术的引入，在带来巨大机遇的同时，也给企业的合规审计工作带来了前所未有的挑战：

- **新型风险的涌现：**除了传统的数据泄露和系统漏洞，企业还必须应对 AI 特有的新型风险，如模型偏见与歧视、对抗性攻击（提示注入、模型窃取）、生成内容合规性，以及复杂的软件供应链安全问题。
- **法规环境的复杂性与动态性：**全球范围内针对 AI 的法规（如中国《生成式人工智能服务安全基本要求》、欧盟 AI 法案）正快速演进，要求企业不仅要保护数据，还要确保算法的透明度、公平性和可解释性，合规门槛显著提高。
- **跨技术栈的审计难度：**AI 应用横跨 MaaS、PaaS、IaaS 多层架构，每一层都有其独特的合规风险点。如何对一个覆盖了第三方 MaaS 服务、自建 PaaS 平台和底层 IaaS 设施的复杂系统进行端到端的、一致性的审计，是一个巨大的技术挑战。
- **持续监控与自动化治理的需求：**云环境和 AI 模型的动态变化特性，使得传统的定期、手动审计模式捉襟见肘。企业亟需建立一套能够持续监控配置变更、实时发现违规行为，并能进行自动化修复的闭环治理体系，以应对敏捷迭代带来的合规风险。

6.3. 具体方案

为应对上述挑战，我们构建一套覆盖“风险识别、策略实施、审计追溯”的全流程合规审计方案。

6.3.1. 分层合规风险与应对

以下以百炼、PAI、ACS 为主要参考分别从 MaaS、PaaS、IaaS 三个层面分别介绍 AI 场景下面临的合规风险和应对。

(1) MaaS 层：百炼

在 MaaS 层，企业主要使用预训练大模型或定制化模型进行推理服务，需要保障推理链路的安全可信。

- **数据传输保护：**基于 PrivateLink 进行私网链接传输，端到端加密传输

- **数据存储保护：**使用 KMS 对依赖的对象存储、云盘等进行存储加密
- **应用安全防护：**使用 WAF、云安全中心对 SQL 注入、命令注入等漏洞进行防护
- **内容安全：**使用 AI 安全护栏对模态违规内容如涉政、色情、暴力等进行监测
- **供应链安全：**使用云安全中心对危险组件进行监测与告警
- **身份与访问控制：**使用 RAM 进行细粒度授权管理、使用 KMS 进行凭证保护
- **Prompt 安全：**使用 AI 安全护栏防御算力滥用与拒绝服务、防御越狱攻击等

(2) PaaS 层：PAI

PAI 依赖的云产品

PAI 产品功能	依赖云产品	依赖说明(使用场景)
Designer	MaxCompute	Designer 中提供上百种基于 MaxCompute 框架实现的阿里自研算法
	OSS	使用深度学习算法组件依赖 OSS 数据源
	Flink	Designer 中提供了几十种基于 Flink 框架实现的阿里自研算法
iTag	OSS	标注数据集的输入和输出，依赖于 OSS 数据源
DSW/DLC	NAS	文件持久化存储需要
	OSS	数据存储需要
AutoML	MaxCompute	超参调优实验可运行在 MaxCompute 上
	OSS	数据存储需要

EAS	API 网关	通过 API 网关的公网调用
	OSS	读取 OSS 的模型文件
	SLS	配置日志写入到 SLS
	VPC	VPC 高速直连
	云监控	服务监控报警
AI 资产	ACR	用户在镜像管理中新建自定义镜像需要

在 PaaS 层，企业通过人工智能平台 PAI 进行一站式的 AI 研发，涵盖数据收集、模型开发、模型训练、模型部署等。需关注以下合规风险：

- **模型可信：**基于人工智能平台 PAI 提供的 ResponsibleAI(RAI)相关能力，贯穿 AI 模型的开发、训练、微调、评估、部署等环节，是保障 AI 模型安全、稳定、公平、符合社会道德的重要方法。
- **数据安全：**使用 HTTPS 对模型部署服务进行安全访问，同时对人工智能平台 PAI 依赖的存储组件如 OSS、NAS 等进行存储加密，实现数据集、模型等数据文件的存储安全。基于阿里云存储产品的高可用，保证用户数据安全可靠。基于存储产品的备份能力进行定期备份，用于异常恢复。
- **基础设施安全：**保障依赖云产品的网络及数据安全。PAI 提供基于 AIMaster 的容错监控能力，能有效进行任务监控、容错判断和资源控制。基于算力健康监测对分布式训练任务的算力资源健康度与性能进行检查。
- **监控与日志：**通过云监控对模型训练及推理服务进行全方位事件和异常监控。基于操作审计和配置审计对云上的操作和资源变更进行持续的跟踪。

(3) IaaS 层：ACS

在 IaaS 层，企业依赖底层基础设施运行 AI 系统。以 ACS 为示例，应重点关注如下合规风险：

- **数据安全：**使用阿里云 KMS 进行 Secret 的落盘加密，基于私网和安全协议进行数据传输，确保敏感数据在存储、传输和处理过程中得到有效的保护，降低数据泄露的风险。
- **网络安全防护：**谨慎开启公网访问，同时基于 WAF、DDoS 等安全产品进行全方位网络安全防护。

- **主机安全加固：**启用云安全中心对主机进行漏洞和镜像安全扫描，提升主机安全防护能力。基于云监控对主机进行运行时水位监控，保障主机的健康运行。
- **身份权限安全：**遵循最小授权原则，合理使用 RAM 与 RRSA，实现 Pod 级别的精细化权限控制。
- **高可用架构：**使用局域级多可用的集群，提升集群的可用性。并在数据面进行有效备份，具备应急情况下快速恢复的能力。

6.3.2. 实施持续合规策略

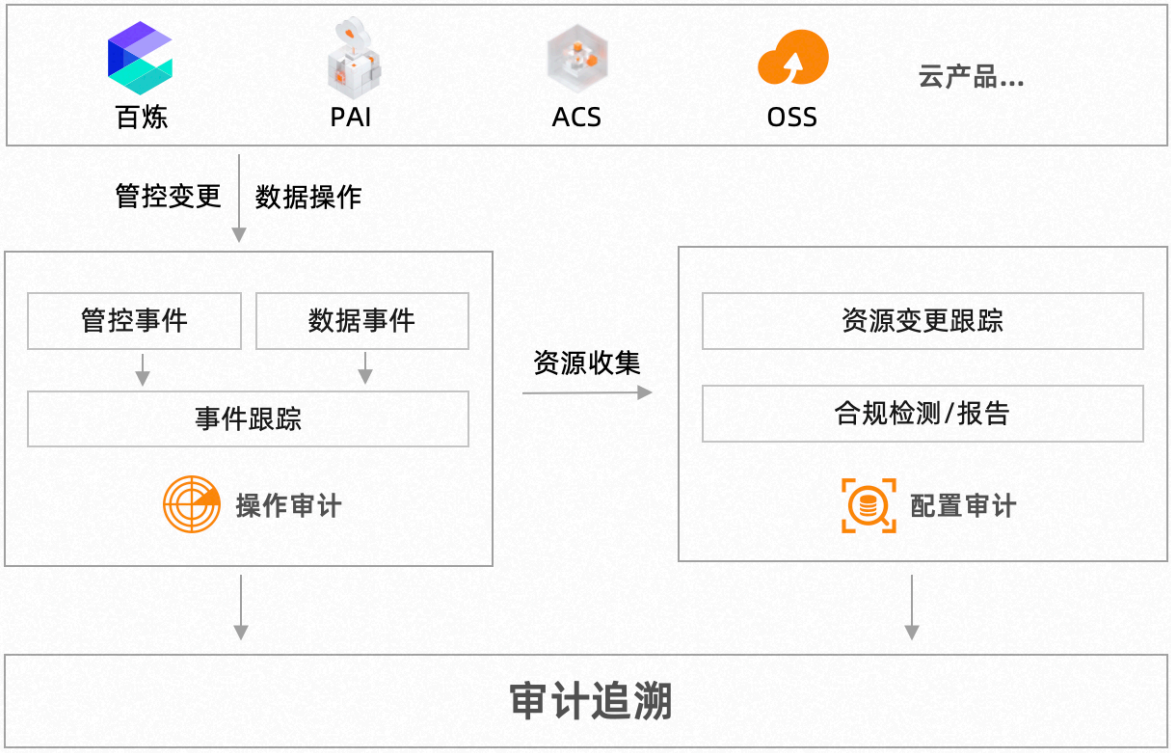
为了帮助客户实现持续的合规监控与治理，阿里云提供了一套覆盖事前、事中和事后的全方位策略能力。具体而言：

1. **事前**：通过对 IaC（基础设施即代码）代码扫描，主动识别潜在的合规风险，确保资源配置在部署前符合规范。在配置审计启用事前规则，基于自动化服务台、资源编排、Terraform 等创建资源时进行主动扫描和防范。
2. **事中**：基于管控策略实时拦截违规操作，防止高风险配置或零容忍问题的发生，保障云上环境的安全性与合规性。
3. **事后**：依托配置审计服务，启用合规包对资源配置进行全方位的持续检测与治理，及时发现并修复不合规资源，确保云上环境始终处于合规状态。

这一全流程治理方案能够有效降低合规风险，助力企业构建安全、稳定且高效的云上架构。

6.3.3. 审计追溯

阿里云通过操作审计与配置审计为企业在云上分别提供面向操作和资源的审计追溯能力。



(1) 操作审计

操作审计聚焦于对用户、系统和服务在云环境中的各类操作行为进行监控和记录，实现对管控面与数据面事件的统一采集、存储与分析。默认提供 90 天的事件查询，企业可以通过设置跟踪对更长时间范围内的事件进行归档和追溯。

- **管控事件**：记录所有通过 OpenAPI 发起的资源管理操作，如创建 ACS 训练集群、修改百炼知识库配置、删除 OSSBucket 等。
- **数据事件**：进一步捕获对数据内容本身的访问与操作行为，例如 OSS 对象的读写、表格存储 OTS 的数据表查询等。

(2) 配置审计

配置审计是一项面向资源的审计服务，可以帮助您实现持续的基础设施的合规监管。

- **资源跟踪**：记录企业云上资源全生命周期的变更记录，并默认提供 10 年的历史查询。
- **合规检测和取证**：提供丰富的检测模板和自定义能力，帮助企业实现一站式的云上合规管理落地。同时提供基于 API 和事件推送的集成能力、基于合规检测的报告能力，满足企业内外部对云上资源合规的举证需求，显著提升审计效率。

6.4. 总结

在 AI 和云计算时代，合规审计不再是事后的、静态的检查，而是一个贯穿始终的、主动的、动态的治理过程。一个有效的合规审计体系能够帮助企业在拥抱技术创新的同时，稳健地管理风险。

核心建议是：

- **采取分层治理模型**：根据 AI 业务所处的平台层级（MaaS/PaaS/IaaS），识别并应对各层独特的合规风险，从数据安全、模型可信到基础设施加固，层层设防。
- **实施全流程合规策略**：建立“事前预防（IaC 扫描）、事中拦截（管控策略）、事后检测（配置审计）”的闭环治理体系，将合规性左移至开发阶段，并确保持续监控。
- **建立不可篡改的审计追溯链**：利用操作审计和配置审计，对所有“操作行为”和“资源状态变更”进行全面、统一的记录和归档，为安全事件调查和合规举证提供坚实的数据基础。

通过构建这样一套“事前有标准、事中有监控、事后可追溯”的自动化审计体系，您可以自信地向监管机构、客户和合作伙伴证明您的 AI 业务是安全、合规和值得信赖的。

7. 运维管理

7.1. 概述

随着 AI 原生应用进入爆发式增长阶段，传统的运维模式已难以应对其在模型训练和推理服务中对效率、稳定性与成本的极致要求。企业迫切需要一套面向 AI 时代的新一代运维体系。本文旨在提供一个从底层基础设施到上层应用的完整可观测性框架，并探讨如何利用 AIOps（智能运维）重塑运维模式，旨在帮助您的 AI 平台“建得快、用得稳、花得明、答得好”，最终实现高效、智能的自动化运维。

7.2. 背景与挑战

在从原型走向生产的落地过程中，AI 应用的运维面临着三大核心挑战，它们共同构成了 AI 时代运维的“新三座大山”。

- **AI 应用的可观测性黑盒**：AI 应用是一个复杂的、由多层次技术构成的体系。其稳定性与一致性难以保障（同一问题多次回答结果迥异），成本透明度低（Token 消耗难以精细化评估），且回答质量与合规性难以自动化审计。这使得运维团队仿佛在面对一个“黑盒”，缺乏有效的、端到端的观测与管控手段。
- **AIOps 的数据驾驭难题：**智能运维（AIOps）是必然趋势，但其有效性高度依赖于海量、异构、实时的可观测数据。这带来了三大难题：异构监控系统导致的“数据孤岛”；数据爆炸性增长带来的“存储与计算瓶颈”；以及让大模型直接处理海量原始数据导致的“算力黑洞”，投入产出比严重失衡。
- **大模型与运维的认知鸿沟：**通用大模型虽然强大，但与专业的运维领域之间存在巨大的认知鸿沟。它听不懂“服务抖动”、“CPU 毛刺”等运维“黑话”，缺乏对系统复杂拓扑关系的认知，并且在进行根因分析时容易因逻辑断链而产生“幻觉”，难以形成可靠的诊断结论。

7.3. 具体方案

7.3.1. AI 应用的可观测

(1) AI 应用生态

当前，AI 应用生态已初步形成由基础模型、开发框架与应用层构成的三层体系，各层协同发展，共同推动 AI 技术从研发走向规模化落地。

第一层：基础大模型（Foundation Models）

以 DeepSeek、Qwen 等为代表的国产大模型正加速追赶国际先进水平，在语言理解、代码生成、多模态等核心能力上持续缩小与 OpenAI、Anthropic（Claude）等头部模型的差距。由于大模型训练需投入海量算力资源（通常依赖万卡级 GPU 集群）和高昂成本，行业参与者高度集中于少数高价值赛道，如通用基座模型、智能驾驶专用模型等，形成了较高的技术与资金壁垒。

第二层：开发框架与工具链

AI 应用开发目前以 Python 为主要编程语言，早期以 LangChain、LlamaIndex 等高代码（pro-code）框架为代表，提供了灵活的模块化能力，支持开发者构建复杂的推理链与 Agent 逻辑。与此同时，多语言生态也在快速演进——例如 Java 领域的 SpringAI、AlibabaCloud 的相关扩展等，正逐步对齐 Python 生态的功能完备性。

此外，低代码/无代码平台（如 Dify、Coze）的兴起，进一步降低了 AI 应用的构建门槛。得益于 AI 应用本身相较于传统微服务更轻量、更聚焦交互与决策的特点，低代码模式尤其适用于快速原型验证与业务场景嵌入。

值得注意的是，完整的 AI 开发栈还需配套基础设施支撑，包括模型上下文协议（MCP）、工具调用（Tools）、向量数据库（VectorDatabase）等，用于实现知识检索、外部系统集成与长期记忆等关键能力。

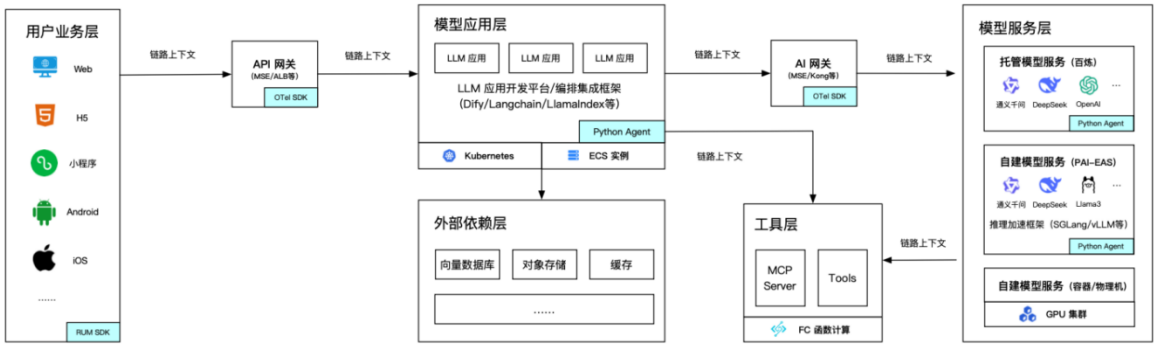
第三层：AI 应用形态

作为生态的最终出口，AI 应用正从单一功能向智能体（Agent）演进。早期以聊天机器人（Chatbot）为主，随后扩展至编程辅助（Copilot）、客服助手、内容生成等垂直场景；如今，具备自主规划、工具调用与多轮交互能力的通用智能体（General-purposeAgent）已成为主流方向，直接面向终端用户提供智能化服务，成为人机协作的核心界面。

这三层架构相互依存、层层递进，共同构成了当前 AI 应用从底层算力到上层体验的完整技术栈。

(2) AI 应用观测的范围

在深入探讨解决方案之前，有必要先厘清典型 AI 应用的系统架构，以便明确观测点的分布与数据关联逻辑。



一个完整的 AI 应用（如智能对话机器人、编程 Copilot 或通用智能体 Agent）并非单一模型的简单调用，而是由用户业务层、模型应用层、外部依赖与工具层、模型服务层及底层基础设施共同构成的多层次技术**体系**。

- **用户业务层：**作为用户直接交互的入口，AI 应用已从早期的聊天机器人（Chatbot）演进为具备自主规划、工具调用与多轮协作能力的通用智能体（General-purposeAgent）。现在大部分的业务入口通常在 APP 或者 web 页面上的对话框，也有一些是车载或者智能音箱的语音交互。这些入口的用户体验是首先需要被关注的。

- **模型应用层**：AI 应用开发以 Python 为主流语言，早期依托 LangChain、LlamaIndex 等高代码（pro-code）框架，支持构建复杂的推理链（ReasoningChain）与多智能体协作逻辑。与此同时，多语言生态加速发展——如 Java 领域的 SpringAI 及阿里云相关扩展，正逐步对齐 Python 生态的能力。低代码/无代码平台（如 Dify、Coze）的兴起进一步降低了开发门槛，尤其契合 AI 应用轻量化、强交互、快迭代的特点，适用于快速原型验证与业务嵌入。

- **外部依赖与工具层**：类比传统微服务中的中间件层，该层为 AI 应用提供运行时所需的公共能力。包括 API 网关、AI 网关、模型上下文协议（MCP）、工具调用（Tools）、向量数据库（VectorDB）等，用于实现不同模型间的调度、知识检索、外部系统集成与长期记忆等核心功能。

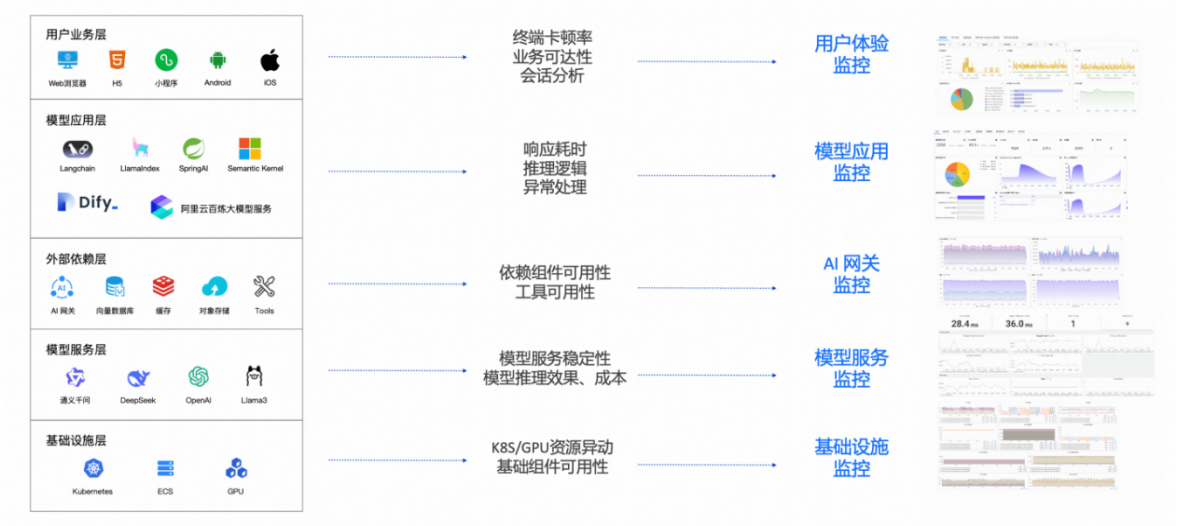
- **模型服务层（Foundation Models）：**作为能力底座，主要有百炼等综合模型服务平台，也有 Qwen、DeepSeek 等开源自建模型服务，提供语言理解、代码生成、多模态交互等大模型能力。这一层对模型的性能（首 Token 延迟、每秒请求数）、成本（Token 数）以及大模型输出效果比较关注。

- **基础设施层：**主要是指通过 PAI、灵骏等提供的 GPU 算力服务，支撑上述大模型等组件的稳定运行。需对底层资源进行精细化监控，尤其是训练场景，对 GPU 算力需求较大，主要集中于通用基础模型、智能驾驶等行业，出现问题的概率非常高。对基础设施的观测主要包括：

    - 智能体所在虚拟机/容器的 CPU、内存、网络指标；
    - 模型推理所依赖的 GPU 集群状态（如显存占用、SM 利用率、功耗）；
    - 存储与网络 I/O 性能等。

7.3.2. AI 应用观测需要具备的能力

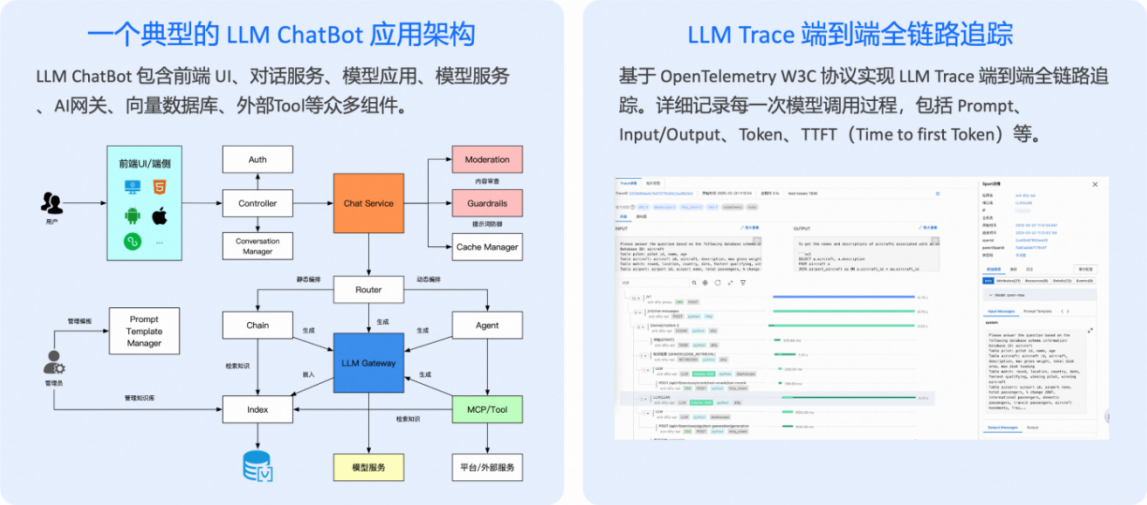
(1) AI 全栈统一监控



在构建 AI 全栈监控体系时，需分层关注不同维度的关键指标：

- **用户层：**重点分析会话过程中是否存在卡顿、中断或响应迟滞等现象，评估其对整体用户体验的影响。
- **应用层：**聚焦于服务响应耗时、异常率及模型推理延迟（如首 Token 时间）等核心性能指标，确保业务逻辑执行的稳定性与效率。
- **网关及依赖组件层：**包括 MCP（ModelContextProtocol）、工具调用（Tools）、向量数据库等中间件，主要监控其服务可用性、调用成功率与错误率，保障系统集成的可靠性。
- **模型服务层：**关注推理效果（如输出准确性、一致性）与资源成本（如 Token 消耗、计算开销），实现性能与经济性的平衡。
- **基础设施层：**监测底层资源利用率（如 CPU、GPU、内存）、缓存命中率及 I/O 吞吐等指标，为上层服务提供稳定的运行环境支撑

(2) 模型调用全链路诊断



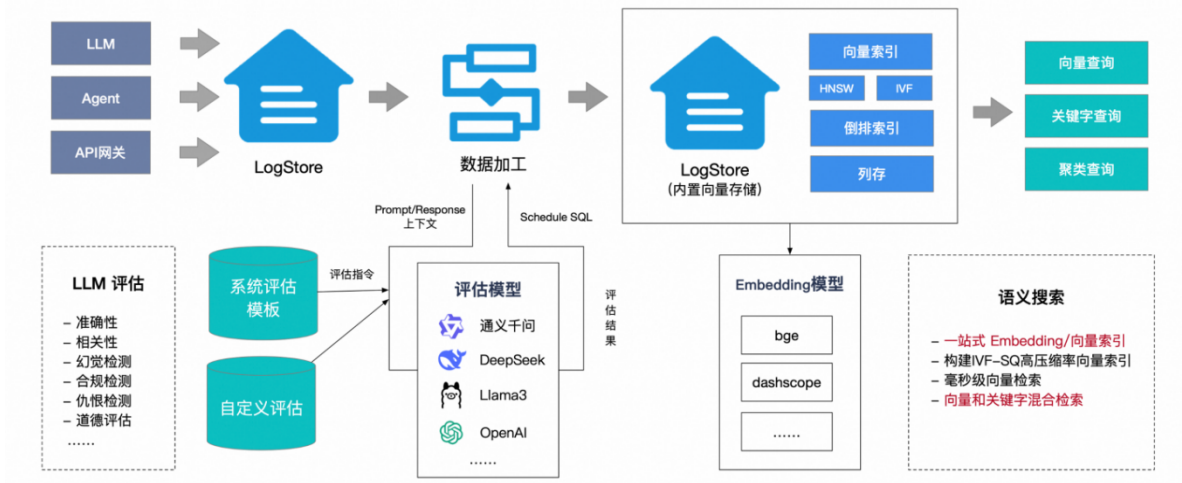
为实现全链路诊断能力，系统需要基于 OpenTelemetry 追踪规范，在从用户终端到模型推理底层的完整调用路径中实施了分层埋点。具体而言，追踪覆盖了用户端、API 网关、AI 应用层、AI 网关以及模型服务层等关键环节。

在实现方式上，部分组件（如 API 网关）采用手动埋点以确保关键流量节点的可观测性；而在 AI 应用层与模型推理层，则通过无侵入式自动埋点技术，无需修改业务代码即可采集运行时数据。无论应用使用 Python 还是 Java 开发，均可通过挂载轻量级可观测探针，自动捕获内部执行细节。

针对 AI 应用特有的复杂逻辑流——包括检索增强生成（RAG）、工具调用（Tools）、多轮模型交互等关键操作——系统在各核心节点均设置了精细化埋点，完整记录每一步的执行耗时、输入输出及状态变化。值得注意的是，模型推理服务本质上也是运行在 Python 环境中的程序，因此同样可部署数据采集探针，深入获取推理过程中的内部指标（如 Token 生成速率、KVCache 使用情况等）。

最终，所有层级的追踪数据需要上报到一个统一的可观测平台，实现从用户请求发起至模型底层推理的端到端调用链贯通，为故障定位、性能分析与体验优化提供坚实的数据基础。

(3) 模型生成结果自动化评估



为保障模型输出质量并支撑持续迭代，需建立一套系统化的评估机制。首先，所有模型调用的输入（Prompt）与输出（Response）均被完整采集并存储于阿里云日志平台。在此基础上，可抽样选取历史记录，结合数据加工流程，引入外部“裁判员模型”对回答质量进行自动化评估。

该评估体系内置多种标准化模板，同时支持未来扩展自定义规则。评估维度涵盖多个关键方面，例如：回答是否包含敏感词、是否存在事实性幻觉（Hallucination）、是否遭受 MCP 投毒攻击等安全与可靠性风险。

具体而言，系统可针对给定提示词及其对应的模型响应，自动判断其是否准确回应了用户问题、内容是否可信、是否存在有害信息等。进一步地，评估结果可被分类与聚类，并打上语义化标签——例如“友善性高”“属于文化类问答”等，从而实现对模型行为的细粒度刻画与趋势分析。

通过建立此类质量基线，每次模型升级或优化均可与历史表现进行对比，确保迭代过程不劣化用户体验，为 AI 应用的稳定演进提供可量化、可追溯的质量保障。

7.3.3. 智能运维 AIOps

(1) 大模型时代带来全新的运维模式

我们看到，AI 正在重塑软件开发，催生了全新的 AICoding 的编程模式。那么，用 AI 简化运维复杂度的智能运维，所谓 AIOperation（AIOps）也必然是时代的趋势。

AIOps 不是新概念。早在 2017 年，Gartner 就提出了这个愿景，希望实现运维领域的自动驾驶。然而，过去的 AIOps 普遍受限于三大瓶颈：僵化的规则引擎、严重的数据孤岛以及高昂的定制化成本，导致其长期难以大规模落地。但如今大模型的出现，为我们突破这些瓶颈带来了曙光，让 AIOps 真正走到了即将突破的临界点。

在大模型时代，要真正实现 AIOps，我们必须解决 2 个棘手的问题：

- **数据的驾驭问题：**当可观测数据从 TB 迈向 PB 甚至 EB 级别时，我们该如何驾驭这片异构、实时的数据海洋，让数据能够为我所用？
- **认知的对齐问题：**我们该如何弥合大模型的通用智能与运维领域的专业知识之间的鸿沟，让 AI 真正“看懂”我们的系统？

(2) 统一可观测数据平台

要解决数据难题，必须有一个强大的平台，一个能支撑好 AIOps 场景的统一可观测数据平台。这个平台需要能够统一支持日志、指标、链路、事件、性能剖析等可观测数据。

- **采集侧，统一数据接入，以打破孤岛：**需要提供覆盖从移动端到基础设施，从传统应用到最新的 AI 应用框架等 200 多种组件的全栈、实时、无侵入的数据接入能力。无论是日志、指标还是链路，每天上报数百 PB 数据，都能被汇入一个平台，为后续 AI 分析提供完整的全局上下文。

- **服务端，统一数据加工与存储，以应对洪流：**需要具备高弹性高可靠架构，能承载每日 EB 级的存储规模，秒级千亿行查询，数百亿行分析，百万时间线计算能力。支持通过丰富、灵活、强大的数据加工能力和存储冷热分层技术，在保障数据完整性的同时，还能降低成本。

另外所有数据都符合需要标准开源协议规范（指标符合 Prometheus 标准，链路符合 OpenTelemetry 标准，事件符合 CloudEvents 标准，使用表格模型均支持 SQL 查询），支持用户自行进行自定义分析或导出。

(3) 使用智能算法降低海量数据的分析维度

有了统一的平台，应该如何解决“算力黑洞”的问题？

AIOps 平台的应该具备计算下推的能力。不要把海量的原始数据喂给大模型，而是将模型的分析意图下推到底层数据引擎去执行。可以通过大量高效的通用算子+可观测数据算子来实现这个需求。可观测算子就像一把锋利的手术刀，专门用于处理特定数据场景。例如指标数据，有丰富的异常检测、预测、聚类、维度下探算法；链路数据，有专门针对链路的异常分析、维度下钻，也有对多调用链构成的拓扑进行构造和对比分析的算子。

由此一来，在实际工作中，大模型只需要扮演智能调度的中枢角色。它接到任务后，会调用这些强大的算子在数据源头进行预处理和关键特征提取。只有高价值信息才会被提交给大模型进行最终的推理和决策。

这种方式，可以将大模型的推理能力和平台强大的数据处理能力完美结合，将 Token 消耗降低 90%以上，让 AIOps 真正变得高效且经济可行。

(4) 基于统一模型重构可观测数据

为了解决这些调整，需要构建一个让 AI 更易于理解的“数字孪生”世界。UModel 统一模型，就是这个数字孪生的核心。它提供了一套观测实体、以及实体关系的定义，帮助我们构建 IT 系统的数字孪生世界，覆盖从用户体验、应用服务、容器到底层基础设施的每一层。

有了这张动态拓扑，排查效率将发生质变。比如，当一个应用报错，我们就可以直接从应用下钻到引发问题的慢 SQL，再到具体的数据库实例；或者，我们可以跨域到它所在的 K8sPod 和 Node 节点上。过去需要在多个系统间的手动跳转、关联的繁琐操作，现在都可以在一个统一的视图内轻松完成。做到这点需要将数据、知识、行动这三个核心要素绑定在一起：

- **数据：**它定义了“是什么”（实体）和“如何连接”（关系）。
- **知识：**将运维领域的专业知识，如黄金指标、健康度、水位、运维手册等，都附着在实体上。这就填平了语义鸿沟，让 AI 能听懂“服务抖动”的真正含义。
- **行动：**将可执行的操作，如回滚、重启、扩容等动作，也都附着在实体上。这就像赋予了 AI 一个工具箱，让它知道面对问题时能对这个实体做些什么。

通过这种方式，UModel 不再仅仅是一张简单的拓扑图，它可以为大模型提供完整的上下文，让大模型进化成一个能理解、会推理、可行动的“数字 SRE”，从而真正开启了 AIOps 的新范式，这是重构可观测的重要一步。

(5) 智能运维助手

有了 UModel，就为智能运维助手（AIOpsAgent）的正式落地扫清了最后的障碍，让大模型可以真正为运维来服务了。运维的交互方式被彻底重塑。用户可以通过自然语言随时召唤它，它能够基于大模型进行自主规划、调用工具、执行、反思，从而解决更多开放和未知的运维难题。这种大模型驱动和自然语言交互的全新形态，整合和升级了运维过程中最核心的四大场景：

- **智能分析：**Agent 变成可观测的数据分析伴侣，无论是复杂的调用链还是火焰图，无论是应用域的问题还是云产品域的问题，都可以直接向它提问，让它为您解读数据，并给出优化建议。
- **智能告警：**Agent 可以帮助配置告警、治理告警，通过调用算法对告警进行收敛降噪（即将升级到 Agent 模式）。
- **根因洞察：**当故障发生时，Agent 能帮助进行根因分析，评估影响，并生成故障总结。
- **智能巡检：**还可以让 Agent 定时对核心业务或集群进行巡检，从“被动救火”转向“主动预防”

智能运维助手作为一个 AIagent 需要能够被集成到各种工具和平台中，所以要能对外提供 MCP 的能力，可以支持通过 OpenAPI 的 MCP 服务调用，这个服务需要具备下面几种能力：

- **基础查询层：**为数据专家提供直接访问原始数据的 API，支持自然语言转 SQL/PromQL。
- **UModel 工具层：**对应拓扑感知和智能洞察，为具备自主规划、工具调用能力的大模型或 Agent 使用，也可以用于 Workflow 编排的 AI 应用场景。提供了基于拓扑和实体的结构化查询能力。此外，支持在将信息喂给大模型前进行预处理，大幅提升分析准确性并降低 Token 消耗。
- **Agent 层：**为用户提供最易用的自然语言可观测接口，专注解决可观测数据分析和问题定位问题，可以与其他管控 MCP 一起集成实现完整的 AIOps 闭环。

7.4. 总结

要实现真正的智能运维并非遥不可及，其核心在于体系化地解决“数据”与“认知”两大难题。这需要我们构建一个三位一体的解决方案：

- **坚实的底座**：一个能够统一接入、计算和存储海量异构数据的可观测平台，这是驾驭数据洪流、打破信息孤岛的基石。
- **智慧的中枢**：通过 UModel 统一模型和智能算法引擎，为数据赋予业务拓扑和领域知识，弥合大模型与专业运维之间的认知鸿沟。
- **全新的交互**：一个支持 MCP 协议、由大模型驱动的 AIOps Agent，它将彻底重塑运维交互模式，将运维工作从繁琐的命令行和仪表盘，带入高效、智能的自然语言对话新范式。

8. 自动化

8.1. 概述

自动化是构建和管理企业级 AI Landing Zone 的核心工程实践。它通过代码来定义、部署和更新基础设施，将手动操作的易错性和低效性转变为可重复、可审计、标准化的流程。本章旨在阐述在构建 AI Landing Zone 时采用自动化的必要性，并重点介绍阿里云推荐的核心方法论——基础设施即代码（IaC），以及官方提供的开箱即用解决方案——Landing Zone Accelerator，帮助您高效、可靠地搭建和运维 AI 云上环境。

8.2. 背景与挑战

随着 AI 业务规模的扩大和复杂性的增加，传统的纯手动控制台操作模式面临着严峻的挑战，这些挑战直接影响到部署速度、系统稳定性和安全合规性：

- **效率与一致性瓶颈**：手动配置环境耗时费力、容易出错，且难以保证多个环境（如开发、测试、生产）之间的一致性。任何微小的配置差异都可能导致难以排查的问题，阻碍敏捷迭代。
- **原生 API 的集成复杂度**：虽然直接调用云厂商 API 提供了极高的灵活性，但这要求团队投入大量的研发资源来处理接口调用、依赖关系、状态管理和错误重试等逻辑，开发门槛高，项目周期长。
- **配置漂移与“黑盒”运维**：在缺乏代码化管理的情况下，线上环境的实际状态往往会因为紧急修复或无记录的变更而偏离其初始设计，形成“配置漂移”。这使得基础设施状态变得不可知，审计困难，系统脆弱。
- **缺乏版本控制与协作**：手动操作无法像代码一样纳入版本控制系统（如 Git），团队成员之间的协作缺少有效的 Code Review 和审批流程，变更记录难以追溯，给安全治理和多人协作带来巨大挑战。

8.3. 具体方案

8.3.1. 自动化构建方式

企业有多种方式可在云上构建 AILandingZone 和管理 AI 相关资源，主要包含以下两种方式：

- 自动化构建
- 手动在阿里云控制台操作

自动化构建不但可以提升效率，而且能够享受到自动化带来的一致性、标准化等能力，减少对人的依赖，提升资源运维管理的韧性，而手动在阿里云控制台操作适用于部分企业在特定发展阶段的选择。

在云上自动化构建 AILandingZone 和 AI 相关资源的常见选择有：

- 调用云提供的原生 OpenAPI 集成到企业内部各类系统
- 采用云提供的 CLI 等命令行工具
- 采用基础设施即代码（IaC）技术

调用原生 OpenAPI 需要感知到 OpenAPI 的复杂性，开发任务较重，但灵活性高，且支持操作的云资源数量多。

采用 CLI 等命令行工具适合运维研发人员日常运维操作或脚本化集成。

采用 IaC 则可以屏蔽直接调用 OpenAPI 带来的复杂性，同时运维研发人员可享受到基础设施即代码带来的收益，比如：状态化管理、代码化管理、开箱即用等。

随着越来越多的企业选择 IaC 来管理云上资源，阿里云提供了通过 IaC 构建 AILandingZone 和 AI 相关资源的自动化解决方案《LandingZoneAccelerator》，加速企业构建 AILandingZone 框架和管理 AI 相关资源。

8.3.2. Landing Zone Accelerator 解决方案

(1) 概述

LandingZoneAccelerator 是一套基于 Terraform 的开源框架，即适用于 LandingZone 也适用于 AILandingZone 框架自动化搭建管理。包含了 AILandingZone 中六大核心模块的自动化构建与持续管理，企业可以下载开源代码到企业本地仓库，并根据企业需要修改对应配置参数来部署符合企业实际需要的云上 AILandingZone 架构和管理云上 AI 相关资源。

(2) 架构

LandingZoneAccelerator 本身是一套基于 Terraform 的云资源管理 IaC 代码，涵盖 LandingZone 六大核心模块的核心能力搭建，但在企业中要采用 IaC，只有代码本身是不够的，还需要配套的企业级的 IaC 管理能力。因此我们为 LandingZoneAccelerator 提供了配套的企业级 IaC 能力，包含以下能力：

- **版本化管理：**与 Git 打通，提供基于 Git 的 IaC 代码版本化管理、多人协作能力
- **Pipeline 支持：**基于 GitPipeline 的 CodeReview 与审批流，同时还可以 Pipeline 中添加 IaC 规范校验能力
- **托管的运行时环境：**基于阿里云自动化服务台的托管式 Terraform 运行环境，免 Terraform 运行环境运维
- **Stack 支持：**阿里云自动化服务台提供了云原生的自研 Stack 能力
- **中心化状态文件管理：**基于 Git+阿里云自动化服务平台提供了基于阿里云 OSS 的中心化 TerraformState 文件管理方式，更适合企业级多人协作开发。

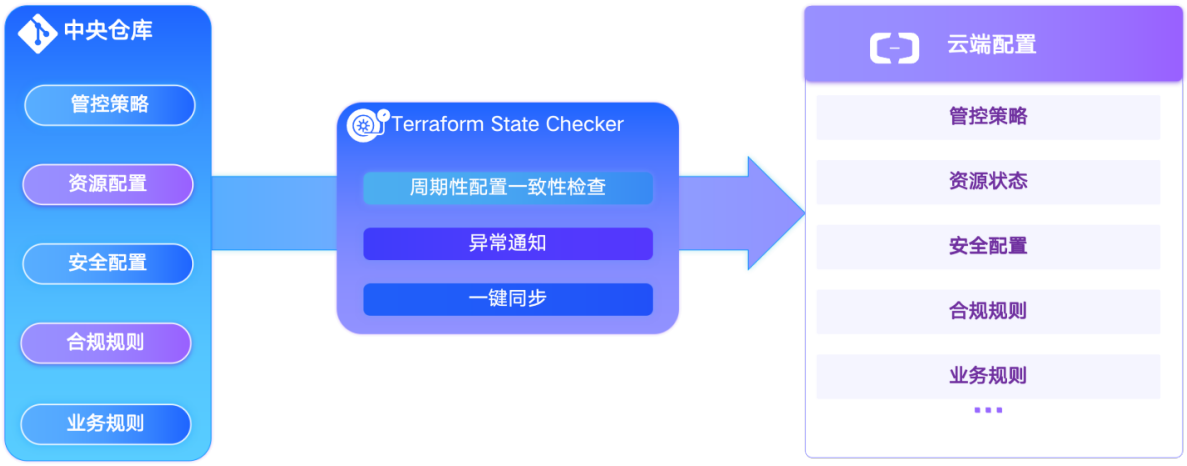
LandingZoneAccelerator 架构非常灵活，基于 LandingZoneAccelerator 源码可与企业其他流程平台、云管平台进行集成，并不受限于默认提供的企业级 IaC 架构。

(3) 扩展性

LandingZoneAccelerator 具备高度可扩展性，LandingZoneAccelerator 作为一个基础核心框架，无法包含所有当前以及未来解决方案中所涉及到的云资源和配置管理，企业可根据自己的需要轻松扩展 LandingZoneAccelerator 的各项能力，包含已有和未来新增的 AILandingZone 解决方案库中各解决方案所提及到的云资源，也支持在 LandingZoneAccelerator 中轻松扩展其他 AI 相关资源的管理，LandingZoneAccelerator 是一套完整的企业级 IaC 架构，包含了完整的 CI/CD、CodeReview、审批流、中心化状态文件管理和多人协作能力，同时也是企业采用 IaC 来管理云上资源的最佳实践架构，完全可以胜任云上所有支持 Terraform 资源的管理任务。

(4) 状态一致性

引入 IaC，最令人担忧的就是云上资源通过其他非 IaC 渠道被修改导致的云上资源状态与 IaC 状态文件不一致，为了解决这个问题，在阿里云自动化服务台中我们也配套提供了 StateChecker 能力，按规则自动持续检查 IaC 资源状态与云上资源状态一致性，一旦出现偏差企业可按需进行修正。



8.4. 总结

自动化并非一个可选项，而是成功实施和长期运维 AI Landing Zone 的必要条件。它将基础设施从一种静态的、需要手动维护的资产，转变为一个动态的、由代码驱动的、可持续演进的系统。

核心建议是：

- **拥抱基础设施即代码（IaC）**：将 IaC 作为云上资源管理的优先范式，以代码化的方式实现标准化、版本化和可审计性。
- **善用 Landing Zone Accelerator**：对于希望系统化、快速搭建 AI Landing Zone 的企业，应充分利用这一开源框架，它不仅提供了经过验证的最佳实践，还内置了企业级的 CI/CD、状态管理和多人协作能力。
- **实现运维闭环**：结合自动化服务台的 State Checker 等能力，解决 IaC 中最棘手的状态一致性问题，形成从部署、监控到修正的自动化运维闭环。

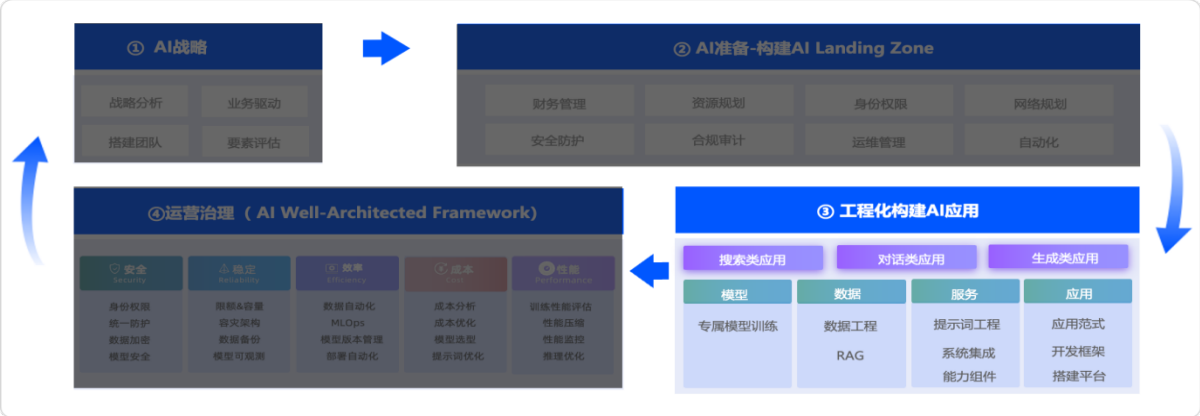
通过将自动化深度融入 AI Landing Zone 的构建与运维流程，企业不仅能大幅提升部署效率和系统韧性，更能为 AI 业务的快速迭代和规模化发展奠定坚实、可靠的工程基础。

第五章．AI 构建

1．综述

人工智能正从通用能力走向深度落地。过去几年，大模型的突破让 AI 展现出前所未有的语言理解与生成能力，但真正推动企业智能化转型的，并非模型本身，而是能够将这些能力嵌入业务流程、解决实际问题的“智能体”。智能体不是简单的聊天机器人，而是具备感知、规划、记忆、调用工具和执行任务能力的自主系统。它能理解用户意图，主动拆解复杂目标，在动态环境中做出决策，并与企业现有系统和数据无缝协同。

当前，智能体已在多个行业初显价值：零售企业用它优化客户服务与库存调度，金融机构借助其完成合规审查与风险分析，医疗机构将其用于病历整理与诊疗辅助，教育和文旅领域则探索个性化内容生成与行程规划。然而，从概念验证到规模化应用，企业仍面临诸多现实障碍。技术路线繁杂、工程集成复杂、私有知识难以有效利用、输出稳定性不足、安全合规要求严苛。这些问题使得许多智能体项目止步于演示阶段，难以真正融入日常运营。



要跨越这一鸿沟，需要一套系统化的方法论和扎实的工程支撑，即第三大篇章：工程化构建 AI 应用。各子章节结构如下：

第二章首先厘清智能体的核心定义与典型场景，明确其区别于传统 AI 应用的关键特征。同时指出企业在推进过程中普遍遭遇的技术、数据与治理挑战，为后续章节提供问题导向。

第三章聚焦架构设计，这是决定智能体能否稳定运行的基础。面对不同业务复杂度，企业需在编排式、目标驱动型和多智能体系统之间做出合理选择。本章提供平台选型的关键考量维度，帮助企业在低代码灵活性与代码可控性之间找到平衡。

第四章深入智能体的“认知内核”，解析其自主决策能力的三大支柱：规划引擎负责任务分解与路径选择，记忆系统支持上下文长期跟踪与经验复用，工具调用则打通 AI 与企业内部系统的连接。这三者协同，使智能体从被动响应转向主动执行。

第五章强调“听懂用户”是所有智能交互的前提。面对模糊、简略甚至矛盾的用户输入，系统需通过意图识别准确把握目标，并借助查询重写技术补全信息、消除歧义。本章对比多种实现路径，指出在企业场景中，往往需要融合规则、检索与模型推理的混合策略，才能兼顾精度与鲁棒性。

第六章将提示词定位为关键工程资产。好的提示不仅是指令，更是业务逻辑的载体。本章倡导将提示词纳入软件工程管理范畴，通过结构化设计、版本控制和自动化测试，确保 AI 输出的一致性与可靠性，避免“一次有效、下次失效”的随意性。

第七章讨论知识增强。大模型的知识是静态且泛化的，而企业需要的是动态、精准、可溯源的信息。通过 RAG（检索增强生成）和实时联网搜索，智能体可在回答前主动查询最新资料，显著降低“幻觉”风险，提升专业可信度。

第八章探讨专属模型定制。当通用能力无法满足细分需求时，微调成为必要手段。本章强调，有效的微调不在于模型规模，而在于高质量、场景对齐的数据和轻量高效的训练策略，使企业能在可控成本下获得差异化 AI 能力。

第九章回归数据本质。无论架构多么先进、模型多么强大，若缺乏持续、高质量的数据供给，智能体终将“失准”。本章提出一套贯穿智能体全生命周期的数据工程体系：从初始训练数据的构建，到 RAG 知识库的动态维护，再到用户反馈的闭环回流与模型迭代。唯有构建这样持续演进的数据机制，智能体才能在实际使用中越用越聪明。

综上，本篇章并非孤立讨论某项技术，而是以企业真实需求为牵引，构建一个从架构设计、核心能力、交互理解、知识管理到数据闭环的完整框架。智能体的未来不在炫技，而在可靠、可管、可用。唯有系统化思考与工程化落地并重，企业才能真正驾驭 AI 智能体，将其转化为可持续的业务价值。

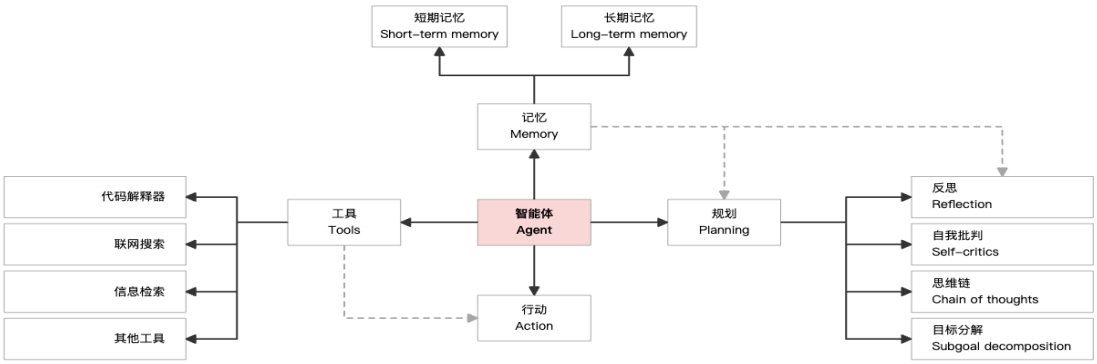
2. 认识 AI 智能体：定义、场景与企业挑战

人工智能正从通用大模型迈向行业深度应用，智能体作为关键载体，正推动企业智能化转型。以大语言模型为核心，智能体融合规划、记忆、工具调用与执行能力，可感知环境、自主决策并完成任务，应用

场景已扩展至客服、知识问答、流程自动化、多模态生成等，覆盖零售、金融、医疗等多个行业。面向企业的智能体，其落地仍面临技术选型复杂、集成难度大、数据稀缺、性能稳定性和合规监管等挑战，需系统化方法论与工程支撑以实现规模化应用。

2.1 什么是 AI 智能体

AI 智能体（Artificial Intelligence Agent，简称 AI Agent）是指具备感知环境、自主决策和执行行动能力的智能系统。它能够根据目标或任务，在动态环境中持续学习、推理、规划并采取适当行动，以实现特定目的，智能体主要依赖规划（Planning）+ 记忆（Memory） + 工具（Tools）+行动（Action）四大核心要素进行工作。其中，LLM 是驱动智能体的认知核心。它融合了对任务的理解、对环境的动态感知以及对历史记忆的调用，通过自主推理形成行动策略，并最终以调用业务 API、工具或知识库等形式，将这些策略转化为对数字世界的具体操作。



2.2 智能体已经在哪些行业真正用起来了

随着大模型（LLM）能力的跃升，Agent 智能体应用已经从“对话助手”演变为能够自主决策、调用工具、调用代码、甚至调用其它 Agent 的复杂系统。智能体在很多场景都已经有了尝试和实际应用，常见场景：

1.智能客服

- **场景描述：**支持多语言交流，满足售前咨询、售后服务、企业知识管理等场景需求。如在售前，客户咨询产品信息时，智能客服迅速响应，基于知识库精准解答疑问，像电子产品客服解答手机参数问题；售后阶段，针对产品使用、故障、退换货等问题，快速提供解决方案。
- **适用客户：**适用于零售、互联网、出行、金融、教育、政务、医疗等行业。

2.知识问答

- **场景描述：**聚焦多源知识整合与动态管理，通过语义理解、多轮交互精准解析用户意图，生成结构化答案。可覆盖企业内外部政策咨询、产品使用等高频问题，并基于问答数据优化知识体系。如企

业员工查询规章制度、项目进展，客服从业务文档找答案，学生或教师查找学习资料、解答疑惑等。能处理 Q&A 快速查询、文档资料库详细解答、知识图谱深入挖掘等不同类型需求。

- **适用客户：**适用于各行业企业构建企业知识库，辅助员工解决工作问题；客服领域提升客户服务效率与满意度；教育领域辅助学生学习和教师教学。如医疗、金融、制造、电商等行业企业，以及学校、培训机构等教育组织。

3.行程助手

- **场景描述：**用户提出旅行需求，如目的地偏好、预算、时间等，行程助手从旅游知识库筛选匹配目的地，介绍景点特色、美食推荐，结合实时交通规划最佳路线，提供航班、高铁等交通信息，还能根据酒店知识库推荐住宿，定制个性化旅行方案。例如用户计划海边度假，行程助手给出三亚、青岛等选择并提供详细旅行信息。
- **适用客户：**适用于互联网、文旅和政府等行业和组织。

4.更多其他场景

- **流程自动化：**报销审批、数据同步、会议安排、跨系统操作等。
- **文档处理：**合同比对、报表生成、票据识别、审计底稿自动生成等。
- **营销内容生成**：自动生成商品文案、促销邮件、短视频脚本等。
- **多语言支持**：理解、生成并处理多种语言的内容，服务于多元语言环境下的用户需求。
- **数据分析与决策支持：**实时监测与预测、自动化报告生成、多模态数据整合。

2.3 企业需要的智能体长什么样

企业级智能体，是面向真实生产环境、深度嵌入业务流程的智能系统，其目标不仅是“能对话”或“会推理”，而是可靠、安全、高效地驱动业务价值，其典型特点如下所示：

- **强任务导向，结果可衡量**：企业级智能体始终围绕具体业务目标（如提升客服响应效率、降低库存成本）设计，强调输出可量化、效果可评估，避免技术炫技，聚焦实际业务成效。
- **深度融合企业专属知识**：在通用大模型基础上，通过知识图谱、RAG（检索增强生成）、领域微调等技术，注入企业特有的产品参数、内部流程、客户数据和行业法规，实现“懂行”的专业语义理解与精准决策。
- **多系统协同与多模态交互能力**：根据场景集成核心业务系统（如 ERP、CRM 等），并支持文本、语音、图像等多种输入输出形式，需要灵活适配复杂、异构的企业 IT 环境与操作规范。
- **安全合规与企业级治理保障**：企业级智能体需要结合权限控制、操作审计、数据脱敏和合规校验机制，满足企业在安全性、稳定性与治理方面的高标准。

- 相较于非企业级 AI 应用，企业级智能体具有鲜明的工程化与业务导向特征。

2.4 为什么很多企业想用智能体却迟迟难以落地

实现真正生产级应用，面临诸多挑战：

- **技术选型复杂**：当前开发框架繁多，涵盖低代码与高代码方案，且模型能力与工具生态快速迭代。企业在面对具体业务场景时，难以高效匹配最适合的技术路径。
- **工程复杂度高**：企业级智能体不仅要准确理解业务需求，还需在授权范围内自主完成“感知–决策–执行–验证”的完整闭环，即能规划任务、调用系统工具、执行操作并反馈结果，真正替代或增强人工流程。这一能力背后涉及多系统集成、权限控制、状态管理与异常处理等复杂工程问题，远超普通对话式 AI 的实现难度。
- **高质量数据获取难**：无论是训练模型还是构建知识应用，都高度依赖结构清晰、标注准确的业务数据。但企业数据往往分散在多个系统、格式杂乱、标注成本高昂，甚至缺乏初始数据（冷启动），严重制约智能体效果。
- **性能与可靠性门槛高**：除需保障系统稳定运行外，部分关键业务（如金融风控、医疗诊断）对结果准确性极为敏感，必须融合 AI 与其他工程手段（如规则引擎等）才能达到实际可用标准。
- **安全合规要求严苛**：企业数据通常涉及商业机密或用户隐私，对数据安全、细粒度权限控制及操作审计有极高要求，通用 AI 方案难以直接满足。

3.智能体系统设计：范式、架构与选型指南

构建 AI 智能体系统需在范式、架构与开发方式间做出关键权衡。编排式智能体基于预定义流程，高可控、强稳定，适用于规则清晰的自动化任务；自主型智能体依赖大模型进行动态推理，灵活应对开放性问题，但存在幻觉、不可解释与安全风险。架构上，单智能体适合简单、线性任务；多智能体通过分工协作（如顺序、分层、群策等模式）处理复杂、跨域场景。开发方式上，低代码快速上线但灵活性受限，高代码支持深度定制但成本更高。最终选型应围绕六大核心问题：场景特征、产品目标、目标用户、团队能力、系统集成需求及数据安全合规要求，确保技术方案与实际场景精准匹配。

3.1 该用哪种开发范式？主流智能体开发范式解析

你的智能体到底该“按剧本走”，还是“自己想办法”？如果任务流程固定、容错率低（比如自动报销审批），是否该牺牲灵活性换取稳定性？如果面对开放性问题（比如客户突发投诉），又能否信任大模型自主决策而不失控？

本节将深入对比编排式与自主型两大开发范式——前者如精密流水线，后者如自由探索者。你将看清它们各自的适用边界、风险代价与典型场景，从而回答那个关键问题：我的业务，到底需要一个“执行者”，还是一个“思考者”？

3.1.1 编排式智能体：确定性流程驱动

编排式智能体是一种以预定义逻辑流程为核心驱动机制的智能系统，旨在通过结构化、可调度的任务序列实现复杂业务流程的自动化执行。该类智能体将目标任务分解为若干有序的子任务单元，并依据显式规则、状态转移机制或工作流引擎进行任务调度与控制流管理，确保各执行环节按既定顺序协调推进。

编排式智能体的核心在于“流程主导、规则驱动”。其运行依赖于预先建模的任务拓扑结构，包括顺序执行、条件分支、循环迭代及异常处理路径等控制逻辑。整个执行过程遵循输入-处理-输出的数据流范式，前一阶段的输出作为后续步骤的输入，形成闭环且可追溯的执行链条。所有可能的状态变迁和决策路径均由开发者在系统设计阶段明确定义，运行时系统仅负责流程解析与节点调度，不涉及自主策略生成或环境探索。

其**优势**在于：

- 高可控性：行为模式完全由预设流程决定，执行路径透明，便于调试、验证与合规审计。
- 强稳定性：在已知场景下表现一致，输出结果可预期，适用于对可靠性要求严苛的生产环境。
- 低开发与维护成本：无需复杂的训练或学习机制，开发门槛相对较低，尤其适合规则明确、边界清晰的业务场景。
- 高性能与低延迟：由于无需实时推理或动态规划，执行效率高，资源消耗可控，适合高频、批量处理任务。

总的来说，编排式智能体代表了一种面向确定性任务的工程化实现方式，其本质是将人类专家知识编码为可执行流程，在保障系统可解释性与稳定性的前提下实现高效自动化。

3.1.2 自主型智能体：目标导向的动态推理与决策

自主型智能体是以大语言模型为核心、以目标为导向的智能系统，能够在不确定环境中通过“感知–规划–行动–反馈”的闭环机制实现动态决策，并持续优化行为以达成目标。其具备高度自治性，不依赖预设流程，而是基于实时状态自主选择策略。

其**核心优势**在于：

- 开放问题求解能力：能够处理目标明确但解决路径未知的复杂任务。
- 复杂任务处理：支持多步推理、跨领域协同与长周期任务执行。
- 强适应性与灵活性：可依据环境变化进行实时调整与自我修正。
- 自主型智能体的高度自治特性也会带来一定**风险**：
- 认知可靠性风险：自主型智能体依赖大语言模型进行感知、推理与决策，而 LLM 本身存在固有的不确定性，包括幻觉、逻辑跳跃、上下文误解等问题。这导致其在任务执行中可能出现：推理偏差、工具误用、目标漂移。

- 行为不可控性与可解释性缺失：由于智能体的决策路径由模型内部隐式推理生成，而非显式编程逻辑控制，其行为过程呈现高度的非线性与黑箱特征，带来如下挑战：决策过程缺乏透明性、决策路径不可追溯。
- 安全与合规风险：智能体自治能力的增强也带来了权限滥用、对抗攻击（提示注入等）与合规盲区的风险。

3.1.3 选择合适的开发范式

编排式智能体适用于任务结构清晰、执行路径固定、操作可标准化的自动化需求。典型应用场景包括但不限于：企业级业务流程自动化、IT 运维自动化（如部署流水线）、数据集成与 ETL 流程、合规审批流、客户服务中的工单处理等。此类系统在强调执行确定性、过程可追溯性和监管合规性的领域具有显著优势。

自主式智能体适用于：目标明确但实现路径开放、环境动态多变、需持续推理与自适应决策的复杂任务场景。典型应用包括但不限于：开放式问题求解（如科研辅助、战略规划）、个性化用户交互（如智能顾问、虚拟助手）、复杂环境下的自主探索（如无人系统决策、游戏 AI）、跨域协同任务（如应急响应调度、供应链动态优化）等。此类系统在面对不确定性高、规则难以预设、需灵活应对的情境中展现出显著优势，尤其适合追求创新性、适应性与智能化水平的前沿应用领域。

3.2 智能体系统怎么搭？单体还是多智能体

一个智能体打天下，还是组建一支“AI 特工队”？当任务从“查天气”升级到“策划一场跨国产品发布会”，单一智能体是否力不从心？而引入多个智能体，又会不会带来协调混乱、成本飙升甚至“内耗”？

本节将为你拆解单智能体的简洁高效与多智能体的协同潜力，包括顺序协作、分层治理、群体决策等典型架构。读完后，你将能判断：你的复杂任务，究竟需要“独行侠”，还是“梦之队”？

3.2.1 单智能体系统：简单高效

单智能体系统（Single-Agent System）是指由单个智能体作为核心决策与执行单元构成的智能系统。该智能体独立地感知环境状态、进行推理决策，并通过执行动作与环境产生交互，以实现预定义的目标或完成特定任务。其核心特点在于：

- 系统中仅存在一个自主决策主体，所有感知、规划和执行功能集中于该单一实体。
- 智能体具备独立的状态评估、目标管理和行为选择能力，无需依赖其他智能体的协作或协调。

根据控制方式，单智能体可表现为编排式或自主式。由于系统内无多智能体交互，故无需处理通信协议、任务分配、冲突消解等多智能体协同问题。

3.2.2 多智能体系统：协同作战

3.2.2.1 多智能体系统的核心机制

多智能体系统（Multi-Agent System）由多个具备自主决策能力的智能体构成，各智能体在共享环境中依据其角色、目标与行为逻辑，通过消息传递、共享记忆或环境交互等方式进行协作、竞争或协商，以协同完成复杂任务并实现系统整体目标。其核心机制在于：系统的整体行为并非由中央控制决定，而是由智能体间的局部互动自组织演化而来。

为有效组织智能体间的协作关系，实践中常采用的典型协作模式如：

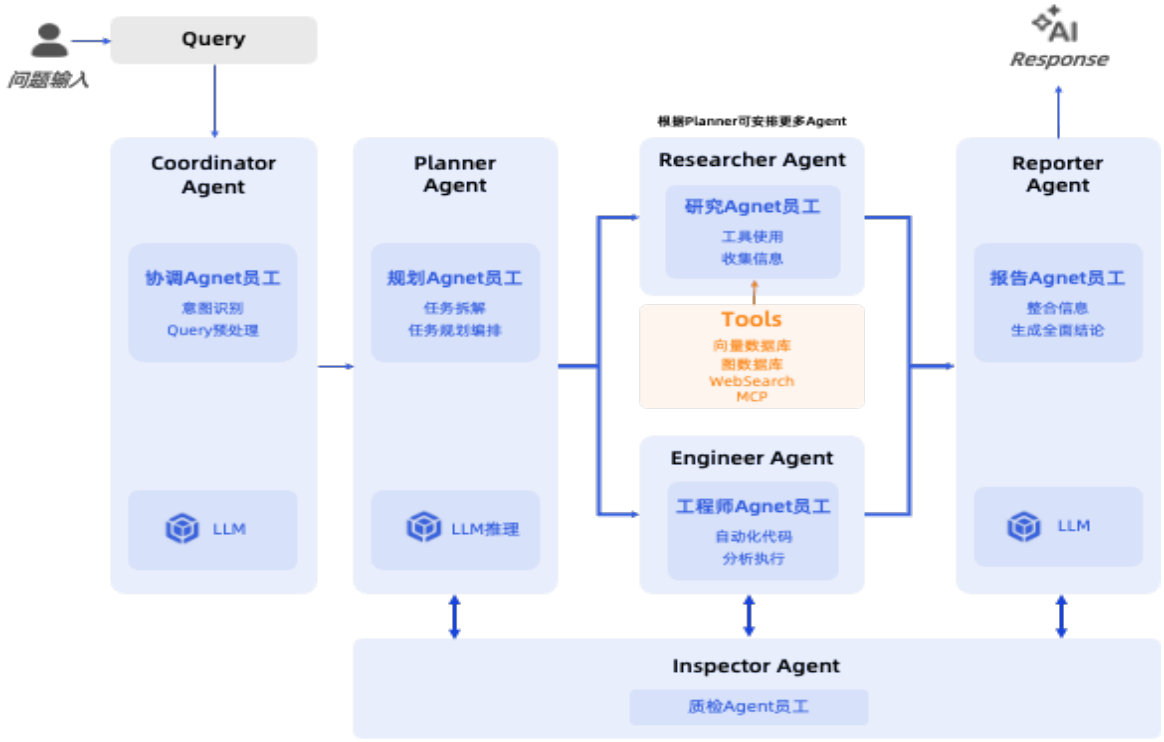
- 顺序模式：多个智能体按照预定义的线性顺序依次执行任务，前一智能体的输出直接作为后一智能体的输入。该模式无需专门的协调者，依赖静态流程编排即可运行。适用于规则明确、流程固定的场景，如自动化文档处理、合规性审查等。
- 分层控制模式：系统中存在一个主智能体（通常称为协调者或调度器），负责接收用户请求、解析任务语义，并将其分解为若干子任务，再根据各专业智能体的能力进行动态分派。每个专业智能体专注于特定领域（如订单查询、退货处理、退款审核等）。该模式适用于结构化程度高、需自适应路由的业务流程自动化，强调明确分工与层级化任务管理。
- 责任转交模式：该模式下各智能体在运行过程中若识别出当前任务超出其能力边界，可主动将控制权转交给其他更合适的智能体（或人类操作员），同时完整传递上下文信息与历史交互记录。该模式不依赖中心化的任务分配机制，而是通过智能体内置的“转交工具”实现去中心化的动态协作。典型应用场景包括客户服务中的技术问题升级、跨域故障诊断等需要灵活求助的交互流程。
- 群策模式：通常由 3 至 7 个智能体组成自治协作团队，通过共享记忆、多轮辩论与迭代优化共同求解高度复杂或模糊的问题。各智能体代表不同专业视角（如市场研究员、工程师、财务分析师等），在持续交互中权衡目标冲突、融合多元知识，最终达成共识性解决方案。该模式模拟人类专家团队的协同机制，能产出高质量、高创造性的结果，但其实现复杂度最高：需设计精密的通信协议、迭代控制逻辑与收敛保障机制，以避免陷入无进展循环、冗余对话或高昂的计算与延迟开销。

3.2.2.2 **案例参考：**DeepResearch 是如何构建多智能体系统的

DeepResearch 是一种基于多智能体协同架构的自动化研究系统，旨在通过大语言模型的高级推理能力、多源异构数据的高效检索以及结构化报告生成，实现复杂研究任务的端到端自动化。该系统通常由多个专业化智能体组成，每个智能体根据其角色配置专属工具集（如信息检索接口、数据库接口、网络爬虫、代码执行器等），并通过预定义的协作协议（如任务分配、信息共享、结果验证）协同完成用户目标。其核心模块包括：

- 智能推理与任务规划层：基于大语言模型，结合思维链、任务分解及分层规划等技术，将复杂研究问题拆解为可执行的子任务序列，并动态调整执行策略。
- 多源信息检索层：集成互联网公开资源、企业私有数据库、第三方 API 及知识图谱等多种数据源，支持语义检索、向量相似性匹配与结构化查询，确保高召回率与高相关性的信息获取。

- 质量保障与对齐控制层：通过双重机制确保输出可靠性，一方面通过内容过滤，利用可信度评分、来源验证与事实一致性检测剔除低质量或不可靠信息； 另一方面通过过程监督与反思，在任务执行中持续监控中间结果与目标的一致性，支持错误检测、回溯修正与最终输出的可信度校验。



以“专利布局分析”为例，一个简单的 DeepResearch 类型智能体的构建过程如下：

第一步：研究框架设计与工具链开发

1. **数据准备：**
    - 内部数据处理：接入企业内部历史专利分析报告（PDF/Word）等数据，构建企业级专利向量知识库，对内外部专利文本进行嵌入并建立向量索引，支持语义检索。
    - 外部权威信源接入：基于工具模块开发多源数据调用工具链（比如：对接 CNIPA（中国国家知识产权局）、 USPTO（美国专利商标局）的开放 API）。
2. **权限策略开发**：设计基于属性的访问控制（ABAC）模型，结合用户角色、数据敏感等级、操作类型进行动态授权；对高风险数据实施端到端保护，包括存储加密、查询脱敏、结果水印及完整操作审计日志。

第二步：研究流程开发与信源溯源实现

目标：通过代码实现动态研究流程，从核心问题拆解为多层级子任务，基于子任务结果动态调用多源数据，最终生成带全链路信源的结构化报告，突破低代码平台的流程模板限制。

1. **研究逻辑编码：**

- 通过高代码方式，增加专属推理、验证规则、反思机制，实现执行过程中的动态调整。
- 设计 prompt 模板：高质量的 prompt 模板，确保后续推理符合行业规范。
- 问题的结构化拆解：通过提示词工程+LLM 或者代码工具实现从核心目标到子任务的层级拆分以及所需调用的工具判断，如① 核心任务 “竞品 xx 一季度在 AI 专利的布局分析”→② 一级子任务 “专利申请量趋势”“技术焦点聚类”“法律状态分布”→③ 二级子任务（如 “技术焦点聚类” 拆解为 “计算机视觉领域关键词提取”“自然语言处理领域关键词提取”），包含子任务依赖关系。
- 动态调用工具，实现子任务的执行。

2. **信源溯源：**在生成每句结论前，强制从已验证的结构化数据源中检索支持证据，并将证据 ID（如专利号、报告段落 ID）作为引用锚点嵌入输出；禁止 LLM 自由编造未检索到的内容。

第三步：报告落地与持续迭代

1. **结构化报告输出：**通过 LLM 模块来实现基本的报告内容输出，可以选择性开发额外的自动化报告模块，将分析结果转化为相关图表（如“专利趋势折线图”“技术关键词聚类热力图”“法律状态分布表”等），并通过代码确保每图表下方标注信源 ID。
2. 在 Web / 移动端部署应用，方便相关的业务人员能够根据自己的场景需要，随时开展研究工作。
3. 持续收集用户对报告的准确性评分（如 “专利分类错误”、“信源遗漏”），调整知识库检索、prompt、插件信息等，优化工作流。

3.2.2.3 选择合适的系统架构

单智能体系统和多智能体系统，二者对比如下：

维度	单智能体	多智能体
任务复杂度	简单，单一控制单元	复杂，多个交互实体
通信机制	无需内部通信	需要通信协议（如消息传递、协商）
推理能力	依赖单一模型能力，易受上下文窗口限制	任务分治，突破单模型能力瓶颈
计算效率	在简单任务中高效	在复杂任务中通过并行提升效率，但通信开销可能增加延迟
开发成本	调试简单，迭代快（适合中小团队）	需设计通信协议/协同机制，开发周期长
可解释性	行为链路易追溯	需额外设计监控和质检模块

若任务目标明确、流程线性、无需多方协商，优选单智能体，譬如：简单自动化任务（简单 FAQ 问答、报表生成）。

而多智能体适用于任务复杂、需分工协作、动态决策或高鲁棒性要求的场景，譬如：分布式决策系统（如智能供应链、金融风控），需要角色分工与动态任务分配（如科研辅助、项目管理）。

3.3 开发方式怎么选？低代码快速上线 vs 高代码深度定制

是追求“今天上线、明天见效”，还是“量身打造、长期可控”？低代码平台让你拖拽几步就跑起一个客服机器人，但遇到特殊业务逻辑时可能捉襟见肘；高代码开发虽灵活强大，却可能让团队陷入数月编码与调试。

“高代码开发”与“低代码编排”的选择需结合应用场景、团队能力、灵活性与效率需求综合决策，以下是几种不同的开发模式实现路径、核心差异及适用场景：

维度	低代码编排	高代码开发	混合开发
适用场景	规则明确、流程固定（客服工单、旅行规划）	复杂逻辑、深度定制（多智能体、科学计算）	核心能力高代码，流程编排低代码
团队能力	业务主导型	资深开发团队（熟悉 Python/TS）	综合能力团队
迭代速度	小时级修改生效	需重新部署，周期较长	核心稳定，流程灵活
长期维护	稳定性较差 升级可能导致工作流失效	稳定性高 代码库易版本控制重构灵活	均衡形态 更灵活的编排

3.4 选型前必须想清楚的六个关键问题

技术再先进，若脱离业务现实，终将沦为昂贵的玩具。为什么有些团队用最前沿的多智能体架构却效果平平？而另一些用简单编排式方案却赢得用户口碑？答案往往不在技术本身，而在问题定义是否清晰。

本节为你梳理出智能体开发前，需系统性评估的六个关键维度，以确保技术选型与业务目标、实施环境及长期演进路径相匹配。

3.4.1 确认场景与任务特征

为什么需要分析该维度：不同任务类型对智能体的能力模型（如规划、记忆、工具调用、反思）有本质差异。若将简单任务用复杂自主 Agent 实现，会导致“过度工程”；反之则会“能力不足”。准确识别任务属性是技术架构选择的前提。

可尝试通过以下问题进行评估：

- 任务目标是否明确？是否有标准答案或流程？
- 是否需要多步骤推理、条件判断或动态调整路径？
- 是否涉及长期上下文依赖（如连续对话中的状态追踪）？
- 是否需调用外部工具完成动作（如联网搜索、调用业务 API）？

选型策略参考：

场景特征	选型策略
任务目标明确、流程线性 （如简单 FAQ 问答、表单填写助手）	优先选择编排式智能体，基于规则+LLM 提示工程实现，结构清晰、可控性强。可使用低代码平台快速搭建。
任务复杂、需多步推理与决策 （如个性化教育辅导、金融投顾）	采用自主式智能体架构，具备长期记忆、自我反思与计划能力（如 ReAct 模式）。推荐使用支持 Agent 框架的高代码平台。
涉及多人协作或多角色分工 （如项目管理协调、跨部门审批流）	构建多智能体系统，通过角色分工与通信机制协同完成任务。可选用支持 Agent 间消息传递的框架。

3.4.2 评估产品目标与定位

为什么分析该维度：评估智能体的产品目标与定位，譬如，是用于验证概念的 POC、初创产品快速上线，还是企业级核心生产系统。这决定了对稳定性、可扩展性、运维能力的要求层级，直接影响技术栈的选择和资源投入。

可尝试通过以下问题进行评估：

- 当前阶段是探索验证、产品孵化，还是正式投产？
- 是否承载关键业务流程或影响营收？
- 是否需要 7×24 小时运行、具备灾备能力？
- 是否计划在未来大规模推广或集成到主系统中？

选型策略参考：

系统定位	选型策略

POC / 概念验证阶段	强调快速迭代与低成本试错。优先选择具备可视化低代码能力的开发平台（如百炼低代码智能体编排），或轻量级开源框架。避免过早投入定制化开发。
初创产品 / MVP 快速上线	平衡速度与可扩展性。核心业务流程代码化以保障速度与稳定性，边缘或变动频繁的分支流程低代码化以提升敏捷性。
企业级核心系统	要求高可用、强监控、易维护。应选择企业级 AI 平台或成熟开发框架（如 AgentScope、百炼）。

3.4.3 确认目标用户

为什么分析该维度：用户群体直接影响交互设计、安全边界、性能要求和部署方式。内部员工能接受复杂操作，而公众用户追求极简体验；外部访问还需考虑并发压力与内容合规。

可尝试通过以下问题进行评估：

- 用户身份：企业员工、B 端客户、C 端消费者？
- 访问方式：网页、App、API 调用？
- 使用频率：高频日常使用 or 低频偶尔触发？
- 对响应速度的要求：毫秒级实时响应 or 秒级可接受？

选型策略参考：

用户类型	选型策略
企业内部用户 （如 HR 助手、财务报销机器人）	可接受一定程度的学习成本，强调与现有系统（OA/ERP）集成。建议采用内网部署+SSO 认证，结合 RAG 增强知识库私有化检索能力，保障数据不出域。
外部终端用户 （如电商导购、医疗咨询机器人）	要求极简交互、高响应速度、7×24 小时稳定运行。常部署于支持自动扩缩容与 CDN 加速的公有云环境；核心在于选用推理高效、具备工具调用与上下文理解能力的大模型（必要时微调），结合模块化智能体设计（如 ReAct），并内嵌多层内容安全与合规审查机制，确保 7×24 小时稳定、安全、可扩展地交付高质量用户体验。

3.4.4 评估团队能力

为什么分析该维度：实际参与智能体开发的人员构成和技术背景，包括是否有专业 AI 工程师、是否是业务人员。团队能力直接决定能否驾驭特定技术栈，以及是选用低代码，还是高代码进行开发。

评估团队构成：

- 是否有熟悉 Python、LLM 原理的 AI 工程师？
- 是否有前端/后端开发人员支持系统集成？
- 业务人员是否参与流程定义？是否具备低代码操作经验？
- 团队是否有 DevOps 或 MLOps 实践经验？

选型策略参考：

开发者类型	选型策略
技术人员主导（高代码能力）	可深度定制 Agent 行为逻辑，选用 AgentScope、LangGraph 等灵活框架，结合训练专属小模型，实现最优性能调优。
业务人员为主（无编程基础）	推荐使用图形化低代码平台，通过拖拽组件配置简单 Agent 流程，适合自动化重复性任务。
混合团队（业务+技术协作）	采用分层协作模式：业务方定义流程与规则，技术人员封装为可复用的 Function Calling 接口；或，核心能力高代码，流程编排低代码。

3.4.5 评估系统集成与扩展能力

为什么分析该维度：智能体是否需要连接外部系统（如 CRM、数据库、ERP）、调用工具（搜索、计算、API），以及是否支持未来功能拓展。绝大多数真实场景下的智能体都不是“孤立思考者”，而是“行动代理人”。缺乏工具调用和系统集成能力的 Agent 只能停留在“聊天”层面。

选型策略参考：

集成需求	选型策略
无需外部系统交互 （如通用知识问答）	可仅依赖大模型自身参数知识，选用纯 LLM 推理方案。但建议仍保留 RAG 接口预留升级空间。
需调用工具或 API （如查天气、订会议室、查询订单）	必须支持 Function Calling / Tool Use 机制。优先选择原生支持该能力的大模型（如通义千问），并搭配 Agent 框架（如 AgentScope）实现插件化管理。
需深度集成企业内部系统 （如对接 SAP、OA、CRM）	建议构建统一 API 网关层，将内部系统抽象为标准化 RESTful 接口，再由 Agent 通过 OAuth 认证以 Function Calling 形式调用。同时，应根据需要优先考虑兼容 MCP 的架构设计，提升工具生态的互操作性、降低集成成本。

未来可能扩展为多 Agent 协作网络	初始设计即应采用松耦合架构，各 Agent 通过 A2A 协议进行通信，便于后续横向扩展。

3.4.6 评估数据安全与合规要求

为什么分析该维度：智能体处理的数据敏感程度及所在行业监管要求，如是否涉及个人隐私、金融信息、医疗记录，是否受 GDPR、网络安全法等约束。一旦发生数据泄露或违规使用，不仅带来法律风险，更会严重损害企业声誉。特别是在金融、政务、医疗等领域，合规是底线。

选型策略参考：

安全与合规等级	选型策略
低敏感度数据 （如公开产品手册、营销文案生成）	可使用第三方公有云大模型 API（如通义千问），享受高性能与低成本优势。
中等敏感度数据 （如客户联系方式、合同摘要）	使用公有云模型 API，可以增加响应的 Safety Guardrail，在 pre/post call 中添加个人敏感信息、企业敏感数据的拦截、脱敏等处理
高敏感度数据 / 强监管行业 （如银行交易记录、患者病历）	建议完全本地化部署（如通义千问开源版），实施严格的访问控制与审计日志。

4.构建智能体核心内核：规划、记忆与工具调用

本章系统阐述了构建 AI 智能体目标驱动决策能力的核心运行机制，聚焦规划与推理引擎、记忆系统设计、工具理解与动态调用三大关键模块，构建支持复杂任务自主执行的智能体“认知架构”。该架构是实现从被动响应到主动决策跃迁的基础，对推动 AI 智能体在开放、动态环境中的实际落地具有决定性意义。

4.1 让智能体学会思考：规划与推理机制

你有没有遇到过这样的情况：给 AI 一个复杂任务（比如“帮我策划一场线上发布会”），它却只能机械地回答零散信息，无法拆解步骤、权衡选项、动态调整策略？如果智能体没有内在的“思考路线图”，它就只是高级检索器，而非真正的决策者。

本节将带你熟悉主流推理框架，并以经典三重循环推理展示如何设计一个高效、自适应的推理规划引擎。

4.1.1 有哪些主流决策方式

大语言模型驱动的智能体系统中，决策机制与推理框架是提升模型复杂任务处理能力的关键。近年来，研究者提出了多种结构化推理方法，以引导模型更系统、可靠地进行多步推理和决策。以下是几种主流的智能体决策机制与推理框架：

- Chain-of-Thought (CoT)：通过逐步推理生成中间步骤，模拟人类线性思考过程。
- Tree of Thoughts (ToT)：将推理建模为树结构，探索多条思路并回溯选择最优路径。
- ReAct：交替进行推理（Thought）与行动（Action），结合外部工具或环境交互。
- Self-Ask：主动分解问题，通过自问自答子问题来解决复杂任务。
- Plan-and-Execute：先制定任务计划，再按步骤执行，支持动态调整。
- Reflexion：执行后自我反思，识别错误并迭代优化策略。
- Graph of Thoughts (GoT)：用图结构表示思想，支持非线性、多路径的信息融合与操作。

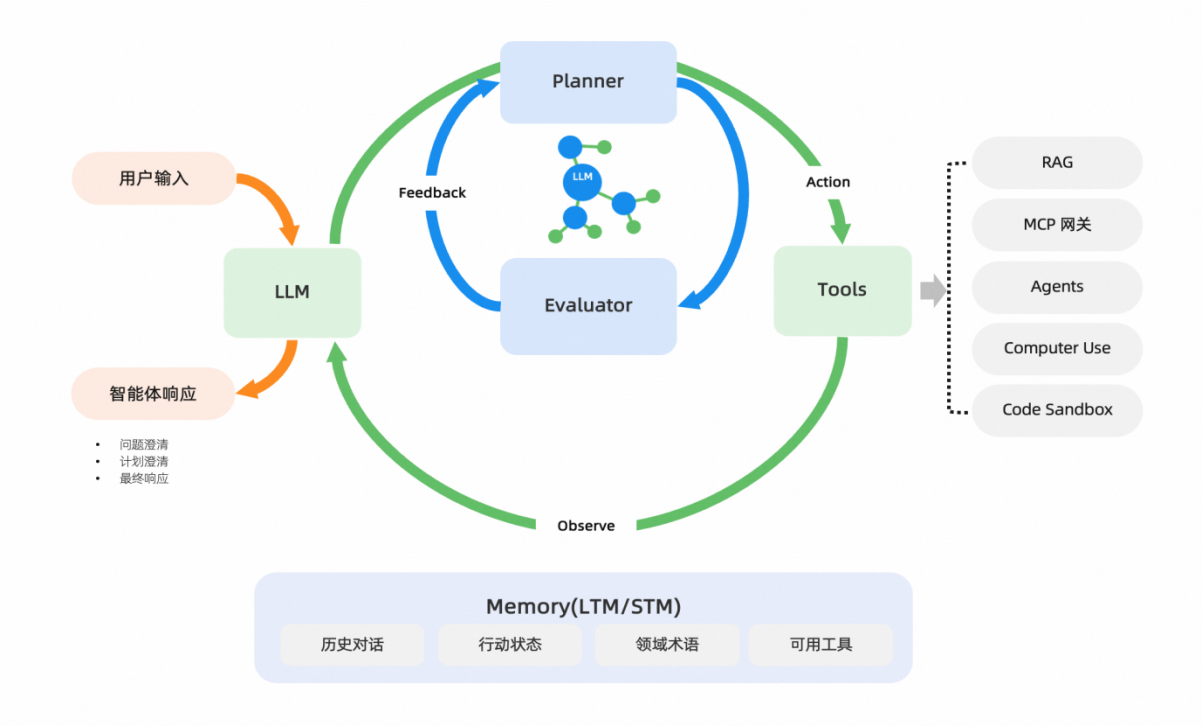
4.1.2 案例参考：用 ReAct 实现经典的三重推理循环

构建智能体的“规划与推理”能力，关键在于设计一个高效、自适应的推理规划引擎。该引擎作为智能体架构中的核心“规划模块”，与大模型“决策大脑”紧密协同，实现类人化的任务理解、策略生成与执行闭环。

传统思维链（CoT）或树状思维（ToT）虽能提升复杂推理能力，但缺乏对执行结果的感知与修正机制。而 ReAct（Reasoning + Acting）框架，可将推理（Thought）、行动（Action）与观察（Observation）深度融合：在每一步中，智能体不仅生成推理逻辑、调用工具执行动作，还基于环境反馈动态调整后续策略。这一机制模拟人类“试错–反思–优化”的认知过程，显著增强系统对不确定性和

动态环境的适应能力。同时，通过引入用户反馈或执行结果作为外部信号，可构建持续学习闭环，实现长期性能演进。

一个成熟的推理规划引擎应有机整合任务分解、多路径探索、工具调用、记忆管理与自我反思等核心能力，以下案例展示了基于 ReAct 的经典三重循环推理框架：



图示：构建多重循环机制，提升推理规划能力的确定性

规划循环：在规划循环中，系统以用户输入的 Query 为驱动，通过规划器（Planner）生成初始执行计划。规划器调用 LLM，结合任务目标与上下文信息制定初步方案。随后，该计划被传递至独立评估模块（Evaluator），由专门的 LLM 对其可行性、完整性及逻辑一致性进行验证。若评估发现计划存在缺陷（如步骤缺失、逻辑错误或资源限制），评估模块将生成详细反馈并返回至规划器。规划器根据反馈优化计划，形成闭环迭代机制，直至计划通过验证，确保其满足用户需求并具备高可靠性。

执行循环：执行循环负责接收规划器输出的最终计划，将其分解为具体操作步骤，并调用相关工具执行任务。推理引擎负责监控执行过程，采集状态与执行结果数据。若执行中出现异常（如工具调用失败、环境变化或意外中断），推理引擎可以基于反馈动态调整计划，以确保任务目标的达成。

用户循环：用户循环作为系统与用户的核心交互机制，旨在让用户掌握任务执行状态并参与关键决策。同时，该循环会主动引导用户澄清模糊问题或追加补充信息，消除潜在歧义，从而提升任务规划与执行的精准性。

4.2 赋予智能体记忆力：从短期缓存到长期经验

想象一下：如果每次对话 AI 都“失忆”，你刚告诉它你的偏好、项目进展或上下文，下一秒它就全忘了。这样的助手能真正帮你完成长期任务吗？记忆不仅是“记住”，更是“有选择地存储、高效地检索、适时地遗忘”。

本节将带你了解智能体记忆的类型，并通过一个记忆管理系统设计示例，展示如何让 AI 拥有“连贯的认知”。

4.2.1 智能体需要哪些类型的记忆

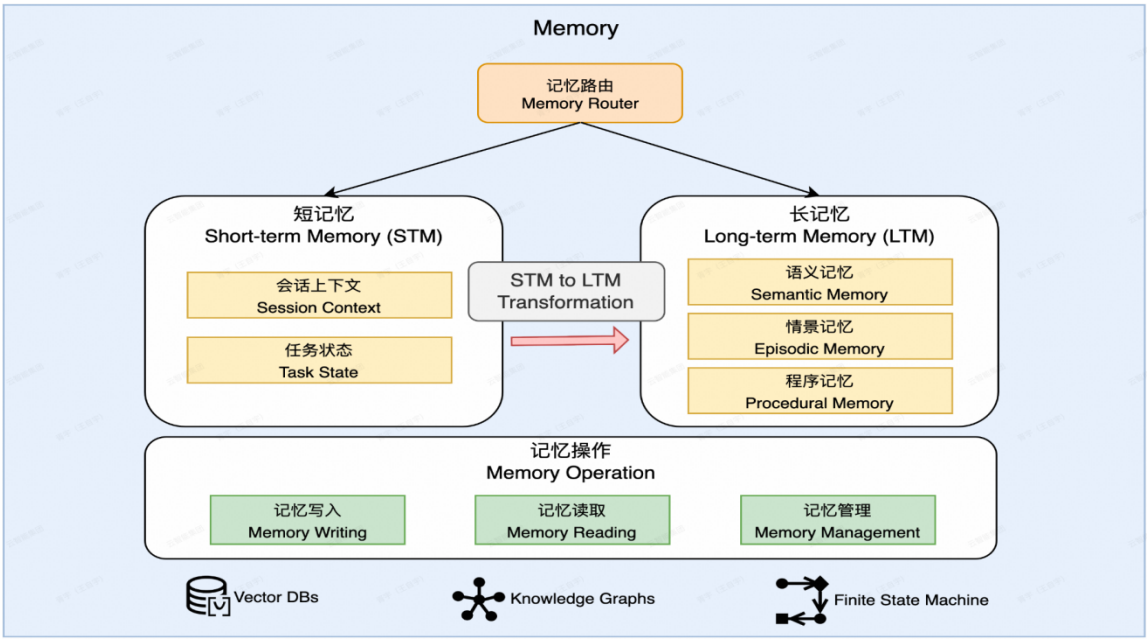
Memory（记忆）记录智能体过往的所见所闻和互动历程，类似于智能体的个人知识库和经验档案。记忆在智能体中对于实现个性化服务、提高任务执行效率、保持对话连贯性、知识积累等方面发挥着核心作用，是提供高质量个性化服务的基础。

记忆的分类借鉴了人类认知心理学中的记忆模型，分为短期记忆和长期记忆。

1. 短期记忆（Short-Term Memory）：也被称作工作记忆（Working Memory），用于存储当前交互中即时相关的信息。帮助智能体跟踪当前任务状态、对话历史或用户请求。短期记忆负责维护对当前步骤的关注，保证在多轮交互中不丢失任务上下文。
2. 长期记忆（Long-Term Memory）：用于保存可能跨越多次会话、任务或时间的重要信息。长期记忆包括语义记忆、情景记忆和程序记忆。
    - 语义记忆（Semantic Memory）：存储随着时间推移而获得的概念性、事实性知识（如领域知识和规则），用于逻辑推理与通用理解。
    - 情景记忆（Episodic Memory）：存储具体交互或环境事件的序列和结果，包含时间、地点和事件细节，用于复盘和经验学习。
    - 程序记忆（Procedural Memory）：是智能体对执行特定任务的流程、步骤或技能的记忆。存储的是“如何做”的知识，而非具体的事实或事件。

4.2.2 案例参考：设计一个清晰的记忆管理系统

整体架构如下图所示：按记忆种类划分为短记忆、长记忆，通过记忆路由模块将请求路由到不同的记忆模块上，并通过后置的记忆操作模块，负责记忆的写入、读取以及管理。



- 记忆路由：根据请求场景确定从短记忆还是长记忆中获取记忆内容。
- 短记忆：短期记忆主要负责维护当前对话或任务中的上下文信息，是智能体保持对话连贯性的基础。短期记忆通常在内存和提示词中存储对话历史数据，从而记录用户的输入信息和智能体响应的完整或摘要信息，并且通常会有上下文窗口大小的限制。
- 长记忆：使其能够存储和检索超出单次会话范围的信息。主要分为语义记忆、情景记忆、程序记忆。
- 记忆操作
  - 记忆写入：信息被智能体感知后，经过 LLM 抽取关键信息后，通过记忆写入操作写入到记忆系统中。
  - 记忆读取：当智能体需要信息进行推理和决策时，记忆读取操作会从记忆系统中提取相关信息以供使用。
  - 记忆管理：通过记忆管理模块，智能体中的记忆可以反思总结生成更高层次的记忆（如将对话信息提炼总结生成高级摘要）、合并冗余的记忆条目、遗忘不重要的记忆。

4.3 教会智能体使用工具：连接现实世界的能力

当 AI 声称“我可以帮你订机票、查天气、发邮件”，却因无法连接外部系统而止步于语言描述。这种“能说不能做”的局限，是否让你感到其行动力严重不足？

本节将介绍工具的分类与接入标准（如 Function Calling、MCP 协议），并演示如何通过 AI 网关统一管理工具调用，让智能体从“纸上谈兵”走向“动手执行”。

4.3.1 工具都有哪些类型

在智能体系统中，工具是指为实现特定目标或完成某项任务而调用的外部功能模块或服务资源。它们是智能体感知环境、获取信息、执行操作和扩展认知边界的“手脚”与“感官”，将大语言模型的推理能力与现实世界的行动能力有效连接。常见的工具类型如下：

- 业务系统 API：封装 CRM、ERP、HRM、审批流等企业核心系统的标准化接口，支持客户管理、资源调度、流程触发等关键业务操作。
- 动态计算执行工具：提供安全隔离的代码沙箱环境，支持运行用户或模型生成的脚本，完成实时数据处理、逻辑推演与自定义计算任务。
- 知识检索增强工具（RAG）：基于检索增强生成机制，动态融合外部知识库与上下文信息，提升回答准确性与领域专业性。
- 环境交互工具：包含 browser-use 与 computer-use 两类。前者支持自动化网页浏览、表单填写与信息抓取；后者可执行本地操作系统指令，实现文件操作、应用调用等桌面级交互。
- 工作流：支持调用预定义或由大模型动态生成的任务流程，实现多步骤、多工具协同的端到端自动化执行。

4.3.2 如何让智能体调用工具

4.3.2.1 通过函数调用快速接入外部工具

Function Calling 并非大语言模型内在的执行能力，而是一种由 API 层实现的结构化输出引导机制，最早由 OpenAI 引入。其核心思想是：通过在提示词中声明一组外部工具的元信息（包括名称、自然语言描述及参数 Schema），引导大模型在推理过程中“决定”是否需要调用某个工具，并生成符合预定义格式的结构化调用请求（通常是 JSON 格式）。该请求由外部运行时系统捕获、解析并执行，执行结果再以结构化形式回填至对话历史，供模型继续生成自然语言响应。

尽管多数平台采用 JSON Schema 描述函数参数结构，但调用请求的具体格式、字段命名、序列化方式存在显著差异。例如，OpenAI 使用 tool_calls 数组，Anthropic 使用 tool_use 消息块，Gemini 则使用 functionCall 对象。这种不一致性导致工具难以跨平台直接复用，需为不同 LLM 适配多套调用逻辑。

Function Calling 的可靠性高度依赖模型是否在训练阶段接触过类似任务。主流闭源模型均通过大量带工具调用标注的数据进行监督微调或强化学习，使其能稳定生成合规的结构化输出。未经此类训练的开源模型（如原始 Llama 3 等）通常无法直接复用该机制，即使提供 Schema，也易输出格式错误或混杂自然语言的内容。

以下为一次典型的 Function Calling **流程**：

1. 工具注册：开发者向 Agent 注册一组可用工具（functions），每个工具包含：name（函数名）、description（自然语言说明）、parameters（JSON Schema 格式的输入规范）。例如注册一个天气查询函数 get_weather(location: str)。
2. Prompt 构造：将所有注册工具的元信息注入系统提示词或作为上下文传递给模型，使其知晓当前可用的能力边界。
3. 模型推理：用户提问：“北京明天会下雨吗？” 模型识别需调用外部数据，输出结构化响应，如：
4. {"tool_calls": [{"function": {"name": "get_weather", "arguments": "{\"location\": \"北京\"}"}}]}
5. 运行时拦截与执行：Agent 框架检测到 tool_calls 字段，解析 JSON 参数，校验类型，并调用本地或远程服务 get_weather("北京")，获取真实天气数据。
6. 结果回填与继续推理：将函数执行结果以 tool_response 形式追加至对话历史，交还模型生成自然语言回复：“北京明天有雨，建议带伞。”
7. 返回用户：最终回答呈现给用户，完成闭环。

尽管 Function Calling 实现简单、上手快，但在复杂系统中暴露出显著**瓶颈**：

- 平台碎片化：各 LLM 平台格式与语义不统一，工具难以跨平台复用。
- 语言/框架锁定：工具绑定特定语言或 Agent 框架，跨语言调用和迁移成本高。
- 扩展性差：缺乏动态发现新工具的能力，新增工具需人工干预提示词。
- 安全与类型风险：参数依赖运行时解析，易出错、被注入，且缺乏语义级安全审查。

4.3.2.2 借助 MCP 协议实现标准化工具交互

Model Context Protocol（MCP）即大模型上下文协议，旨在解决 LLM 工具调用的碎片化问题，建立大语言模型与外部工具、数据源之间的通用、安全、可互操作的通信桥梁。Anthropic 在其 Claude Desktop 应用中率先集成 MCP 作为工具调用方案，推动了该协议在开发者生态中的早期落地。

MCP 采用客户端-服务器（Client-Server）架构，通过标准化的消息格式与交互语义，使智能体能够动态发现并安全调用各类专用服务，而无需在提示词或运行时中硬编码集成逻辑。其核心理念是：将智能体与其可用工具彻底解耦，通过统一协议实现“即插即用”的工具生态，相当于为所有 AI 工具提供了一个通用的“USB-C 接口”。

MCP 由三个核心角色构成：

- Host（宿主应用）：运行 LLM 智能体的应用程序，负责管理对话状态、触发工具调用决策，并协调与外部能力的交互。Host 内嵌一个或多个 MCP Client 实例。典型 Host 包括 Cursor、Claude Desktop 等 AI 增强型 IDE 或智能体平台。

- Client（客户端）：轻量级模块，运行在 Host 内部，代表 Host 与远端 MCP Server 建立连接。一个 Host 可拥有多个 Client 实例，以连接不同 Server。
- Server（服务端）：遵循 MCP 协议暴露能力的服务进程。

MCP 支持 STDIO、SSE（Server-Sent Events）、Streamable Http 等传输模式。其中，STDIO 适用于本地命令行环境或进程间通信来访问本地文件系统、数据库等资源。SSE 适用于 Web 环境中需要实时接收模型输出（如流式文本生成）的场景。Streamable Http 则侧重高灵活性和多路复用，适用于需要实时交互或者需要双向通信等场景。

完整交互流程如下：

1. 初始化阶段
   A. Client 发送 initialize 请求，声明自身支持的协议版本与能力。
   B. Server 返回 initialize 响应，包含其能力清单、服务器元信息及可选认证要求。
   C. 双方完成能力协商，建立有效连接。
2. 能力发现阶段
   A. Client 查询 Server 的工具/资源/提示词模版列表（tools/list 或 resources/list 或 prompts/list）。
   B. Server 返回所有可用工具/资源/提示词模版的元数据（含名称、描述、JSON Schema 等）
   C. Client 缓存这些信息，供 Host 中的智能体进行选择。
3. 能力调用阶段
   A. Host 决定调用某工具后，通过 Client 发送 tools/call 请求，指定工具名与参数。
   B. Server 执行实际逻辑（可能调用外部 API、读取文件等）。
   C. （可选）Server 发送 progress 消息提供执行反馈。
   D. Server 返回最终结果（成功或错误）。
4. 连接终止：双方可通过 shutdown 消息优雅关闭连接，或由任一方主动断开。

4.3.2.3 如何选择最适合你的工具接入方案

Function Call 与 MCP 对比如下所示：

维度	Function Call	MCP（Model Context Protocol）
设计定位	模型内嵌的工具调用机制，用于单次推理中触发外部函数 。	标准化协议，定义模型与外部工具/服务间的通用交互接口。
标准化程度	厂商私有，格式碎片化。	开放协议，统一接口与交互规范。
多模型适配	需针对不同模型定制实现，兼容性差。	模型与工具解耦，任意兼容 MCP 模型可复用工具。
跨语言调用	模型与工具通常同语言绑定，跨语言调用需封装。	工具为独立服务，协议通信天然语言无关。
发现与扩展	工具通过静态声明，新增需修改提示或代码，无动态发现。	支持服务注册与动态发现。
性能开销	单次调用延迟低，但多轮需多次推理，累积延迟高。	单次有网络开销，但支持批处理/异步，整体流程更可控。

Function Calling 在工具集稳定、调用逻辑简单、变更频率低的场景下可胜任生产环境，已被广泛应用于客服机器人、BI 助手、内部运维自动化等企业级系统。然而，在需要动态扩展工具、跨模型协作、强类型保证、复杂状态管理或多步骤规划的系统中，应考虑采用标准化工具协议（如 MCP）或支持 MCP 的高级 Agent 架构，以实现跨模型、跨语言、跨运行时的工具互操作。

总的来说，Function Calling 更适合作为轻量级、临时性或原型验证阶段的集成手段。在构建多智能体协作或高频迭代的生产系统时，建议采用 MCP 作为统一的工具集成标准，通过其标准化的发现、调用与上下文传递机制，替代碎片化的原生 Function Calling，从而提升系统的可维护性、安全性与扩展性。

4.3.2.4 案例参考：通过 AI 网关统一管理所有工具

阿里云 API 网关提供的 Al 网关能力支持 MCP 服务的流量代理，且能够在零代码改造的基础上将已有的 HTTP 服务转化为 MCP 服务。所有流经 MCP 服务的请求均由 Al 网关统一接入和处理，安全防护、流量管理、可观测等能力全部收敛在网关侧，让开发者能够专注于核心业务逻辑的实现。

这里以 MCP 服务接入为例，介绍如何将工具进行注册。根据业务需求的不同，可以分为三种场景：

场景 1：企业已有存量 HTTP 服务，希望将存量 HTTP 服务无侵入式转换为 MCP 服务

针对企业中已有的存量 HTTP 服务，平台提供“零代码改造”的 HTTP→MCP 协议转换能力，帮助企业快速升级现有架构，实现与 AI 网关、MCP 生态系统的高效对接。以该种方式接入时，AI 网关可以提供协议自动转换能力，无需修改原有服务逻辑，即可将传统 HTTP 接口无缝转换为符合 MCP 协议的接口，它支持 SSE、Streamable HTTP 两种主流 MCP 协议类型，支持 HTTPS 服务识别：区分原有服务是否启用 HTTPS 协议，确保服务安全接入。另外，如果企业本身的 HTTP 服务已经在 MSE Nacos 中注册，网关

还可以实现自动化服务注册。同时还支持开启 KMS 密钥服务进行加解密处理，增强敏感数据的安全性。企业可选择已有 KMS 密钥，或提前购买并配置软件密钥实例。

场景 2：企业已有开发完成的 MCP Server，希望直接注册并供 Agent 调用

针对企业中已构建完成的 MCP 服务，平台提供标准化、开箱即用的 MCP 服务接入能力，支持开发者在不修改服务逻辑的前提下，快速将服务注册至 AI 网关，并开放给 Agent 进行调用。这里提供两种便捷的服务注册方式，适配企业不同服务治理现状。

A. 使用已注册服务：若服务已在 MSE Nacos 中注册，可直接引用，实现快速接入。

B. 新建服务配置：用户可自定义输入 IP/域名、端口及访问路径，支持灵活部署与管理。

用户通过控制台完成相关配置后，可一键保存并发布服务，实现从开发到上线的无缝衔接，显著提升服务交付效率，降低运维复杂度。通过该标准化接入方式，企业能够快速将自有 MCP 服务集成至 AI 网关生态，充分发挥 MCP 协议优势，支撑 AI Agent 的高效调用与协作。

场景 3：企业没有 MCP Server，也没有存量的 HTTP 服务

在这种场景下，可以通过使用 Spring AI Alibaba 框架，或者 Nacos MCP Wrapper Python 开发 MCP Server, 这种方式支持在 MCP Server 启动后动态注册至 Nacos，并对齐进统一管理，具备以下能力：

A. 服务动态管理：通过 MCP 服务列表增删改查服务信息。

B. 描述动态生效：工具描述、参数定义等元信息支持运行时热更新，无需重启服务。

C. MCP Server Tools 动态开关：支持 MCP Server 服务 Tools 运行时动态开启和关闭，无需重启服务。

D. 全链路集成：服务注册信息自动同步至 Nacos 配置中心与服务发现模块，适配 AI Agent 调用需求。

以下是对三种 MCP 服务接入方式的特点总结：

接入方式	特点	是否需要代码改造	安全支持	服务注册方式	动态管理能力
场景 1：HTTP → MCP 协议转换（零代码）	零代码改造，自动将 HTTP 转换为 MCP	❌ 不需要	支持 HTTPS 和 KMS 加密	自动识别 Nacos 注册信息	❌ 不支持动态管理
场景 2：已有 MCP Server 注册（标准化接入）	快速注册已有的 MCP 服务，无需改动服务逻辑	❌ 不需要	可选启用 KMS 加密	手动或引用 Nacos 注册信息	❌ 不支持动态管理
场景 3：使用 Spring AI Alibaba / Python Wrapper 开发	基于框架开发，具备丰富的动态管理能力	✅ 需要开发	支持 HTTPS 和 KMS 加密	自动注册到 Nacos	✅ 支持运行时热更新

以下是适用场景建议，帮助企业根据自身技术现状、业务需求和开发能力选择最合适的方式进行 AI 网关对接：

企业现状 / 需求	推荐接入方式
有存量 HTTP 服务，不想改代码，想快速接入 AI 网关	场景 1：HTTP → MCP 协议转换
有已完成的 MCP Server，希望快速上线并供 Agent 使用	场景 2：已有 MCP Server 注册
正在开发新服务，需支持动态管理、热更新、Agent 调用	场景 3：Spring AI Alibaba / Python Wrapper 开发

5.提升输入理解能力：意图识别与查询优化

本章聚焦提升智能系统对用户输入的理解能力，核心在于意图识别与查询重写两项技术。意图识别是理解用户目标的基础：在单体系统中驱动对话与响应，在多智能体协作中支持任务拆解与调度。其技术从人工规则发展到机器学习分类器，再到当前基于大语言模型的上下文感知理解，显著提升了准确性和适应性。针对企业场景中意图类别多、边界模糊、表达多样等问题，本章对比了向量检索匹配、大模型提示推理和垂直领域微调三种实现方式，强调实际应用需融合规则、实体识别与上下文管理。查询重写则通过指代消解、信息补全、歧义消除和复杂查询拆分，将原始输入转化为结构清晰、语义完整的标准形式，提升意图识别与后续执行效果。二者共同构成系统“听懂用户、理清需求”的关键基础。

5.1 意图识别

用户输入“把上个月的数据发我一下”，系统却不知该调销售报表、日志数据还是财务汇总——因为“数据”背后藏着未明说的业务意图。在企业场景中，用户表达千变万化，意图边界模糊交错，仅靠关键词匹配或简单分类器早已力不从心。

本节将带你梳理意图识别的技术演进路径，对比向量检索、大模型推理与领域微调的优劣，并通过一个混合方案，展示如何在复杂环境中精准“读懂”用户的真实目标。

5.1.1 什么是意图识别

意图识别是指通过自然语言理解技术，从用户输入中自动推断其背后的目标、需求或期望行为的过程。其核心在于将非结构化的用户表达映射到预定义或动态生成的语义意图类别上。

在智能体系统中，意图识别是对话管理的关键前置模块，用于驱动后续的槽位填充、对话策略选择与响应生成，确保系统准确理解并满足用户在单轮或多轮交互中的真实目标。

5.1.2 从规则到大模型：意图识别是如何一步步变“聪明”的

有关意图理解和识别的技术演进大致可粗分为三个阶段：

1. 基于规则匹配：早期系统依赖人工构建的规则（如正则表达式、关键词匹配或有限状态机）。在封闭域和固定场景下表现尚可，但泛化能力弱、维护成本高，难以应对语言的多样性与歧义性。

2. 基于传统机器学习：采用分类器（如 SVM、朴素贝叶斯、逻辑回归）结合手工特征（如 TF-IDF、n-gram），具备良好可解释性与训练效率，在小规模、领域明确的数据上有效，但严重依赖特征工程，难以建模复杂语义关系与长距离依赖。

3. 基于深度学习模型：以 CNN、RNN、LSTM 等神经网络显著提升意图识别性能；预训练语言模型（如 BERT）通过上下文感知的语义表示，大幅降低对人工特征的依赖；大语言模型（进一步展现出强大的零样本/少样本意图理解能力，可通过提示（prompting）直接判别用户意图，并支持多轮对话中的动态意图追踪。

在特定场景中，传统方法仍具实用价值。而在智能体应用中，主流方案已转向深度学习模型，特别是基于结合大语言模型的意图理解和识别。

5.1.3 企业场景下意图识别的典型挑战

针对不同复杂度的企业智能体应用场景中，意图识别常见的挑战如下：

1. **意图类别规模庞大：**企业级场景中，意图类别可能达到数百甚至上千个，远超简单意图节点的处理能力。这种大规模意图分类对系统的扩展性和性能提出了更高的要求。
2. **复杂层次和优先级：**实际业务中的意图并非单一层次的平铺结构，而是可能包含多层次、多维度的嵌套关系。不同意图之间又存在不同的处理优先级。
3. **意图边界的模糊性：**某些意图之间可能存在较高的语义相似度，导致传统的单一技术手段（如关键词匹配或通用模型分类）难以实现高精度的识别。
4. **多样化的表达方式：**用户对同一意图的表达方式可能多种多样，包括大量的同义词、近义词以及上下文相关的变体。这种多样性增加了意图识别的复杂性。
5. **性能与管理复杂性：**即使通过多次使用简单意图节点逐步拆分意图分类，也会显著增加整体意图识别的时长，并导致意图管理的复杂性急剧上升，难以适应生产环境的需求。

5.1.4 如何选择最适合的意图识别方案

结合大语言模型，在智能体应用中对于意图识别常见有如下方式，它们在响应速度、准确性、成本上各不相同，针对不同的应用场景，企业需要灵活选择。

1. 基于向量检索：
    A. 特点：基于文本向量化技术，将意图名称、描述及其样例集转化为高维向量表示。用户输入 query 时，通过向量检索匹配语义相似度最高的样例，并以其归属的意图为命中结果。该方案无需复杂的模型推理，直接依赖预计算的向量索引进行快速检索。
    B. 适用场景：适用于语义边界清晰、指令明确且对响应延迟敏感的场景（如简单命令式交互），仅需提供高质量样例即可快速部署。
2. 基于大语言模型的意图判断：
    A. 特点：利用基础大模型（如 Qwen 等）结合 Prompt 工程实现意图识别。通过零样本或少样本推理，依靠模型的强大上下文理解能力完成分类任务。支持分层意图识别，可递进式下钻至目标叶子意图，但层级不建议超过 3 层。
    B. 适用场景：适用于用户表达丰富，意图数量有限、标注样本较少的场景，需提供精准的意图定义与少量示例，借助模型上下文理解能力完成识别。

3. 基于领域微调模型的意图判断：

   A. 特点：使用小尺寸模型进行领域微调，通常采用 LoRA 等参数高效微调方法。微调后形成领域专属模型，能够在特定领域内提供高精度的意图识别服务。

   B. 适用场景：适用于对准确率、领域知识覆盖和响应延迟均有严苛要求的专业场景（如金融、医疗、法律等），需投入一定标注与训练成本以换取长期稳定性和系统效率。

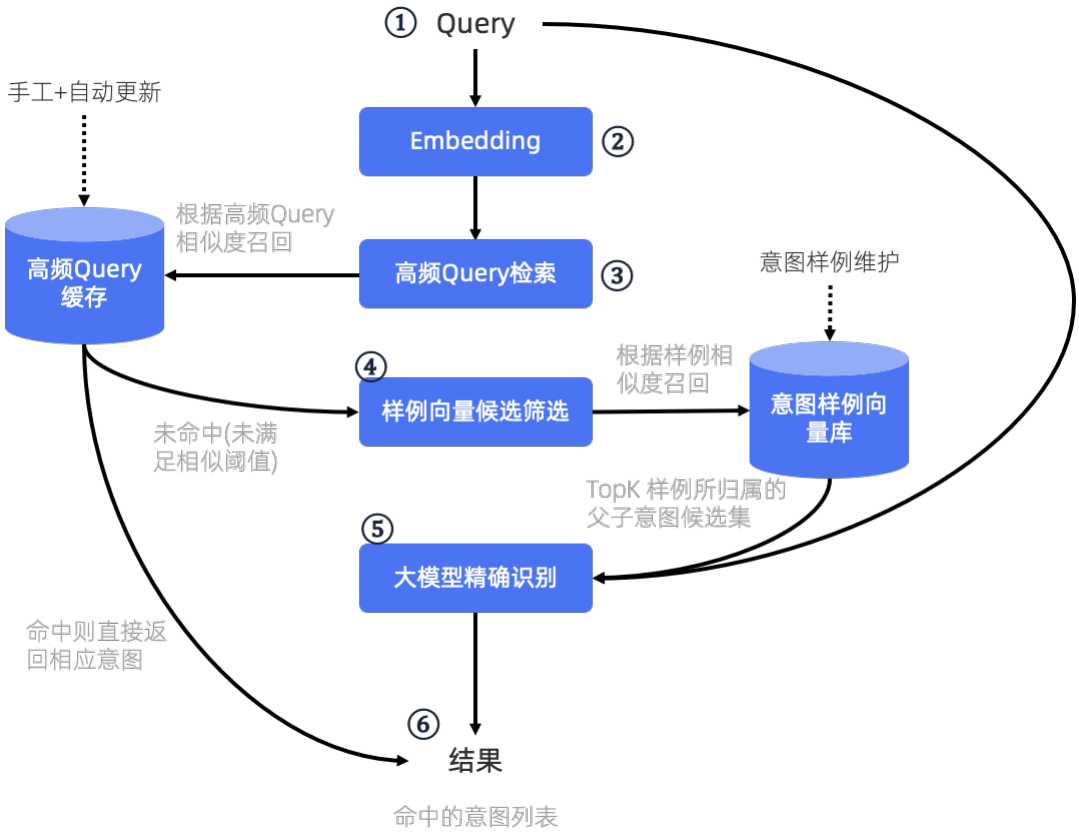
下表展示了不同方式在响应速度、准确性、成本上的差异：

	核心技术	适用场景	速度	准确性	成本
基于向量检索的意图判断	向量检索	意图数量多、样例丰富、要求低延迟	快速	中等	低
基于大语言模型的意图判断	基础大模型 + Prompt 推理	意图数量少、样例有限、追求高准确性	中等	高	中等
基于领域微调模型的意图判断	小尺寸模型 + LoRA 微调	垂直领域、高精度需求、接受较高开发成本	快速	很高	高

需指出的是，上述意图识别方式本质上属于典型的意图分类任务。在企业实际应用中，为满足多样化的业务需求，完整的意图识别方案通常需融合多项技术：包括命名实体识别、Query 重写（用于歧义消除、指代消解、查询补全等）、检索增强生成（用于注入领域知识或动态规则）等。在特定场景下，亦常结合基于规则的意图识别模块进行前置过滤，以提升整体准确率与稳定性。此外，依据具体业务流程，系统还可能集成槽位提取等环节，实现端到端的语义理解与任务执行。

5.1.5 案例参考：构建一个兼顾准确率与灵活性的混合识别系统

下图展示了采用“向量检索 + 大模型判断”的混合意图识别架构：高频查询通过向量检索快速匹配意图，低频或未命中查询由大模型兜底判别，兼顾系统性能与识别准确率。



1. Query：Query 已经过预处理（如问题分解、改写）。
2. Embedding：Query 编码为语义向量，用于后续相似度计算。
3. 高频 Query 检索：在高频 Query 缓存中进行近似匹配，若相似度超过阈值（如 95%），则直接返回对应意图，实现高效响应；未命中则进入下一级。
4. 样例向量候选筛选：基于维护的意图触发样例向量库，通过向量相似度检索 Top-K 候选意图，缩小判断范围。
5. 大模型精确识别：结合 Query 与候选意图集，利用大模型进行语义理解与意图判别，输出最优意图及置信度。
6. 结果：返回最终意图列表，包含意图名称、置信度及推理路径等可解释信息。

5.2 查询重写

当用户说“它昨天又出问题了”，而“它”指代不明、“昨天”缺乏时间基准、“出问题”过于笼统——这样的输入，AI 该如何执行？原始查询往往残缺、模糊甚至自相矛盾，直接用于意图识别或工具调用，极易导致错误。

本节将深入查询重写的核心能力：如何通过指代消解、信息补全、歧义澄清与复杂句拆分，将“人话”转化为机器可执行的清晰指令，并通过基于大模型提示工程的实践案例，展示如何让系统真正“听懂弦外之音”。

5.2.1 什么是查询重写

用户原始查询常因噪声、歧义或意图偏差而影响下游任务的效能，典型问题包括：

- 专业术语混杂：领域缩写或术语（如“EBITDA margin”）需准确解析与标准化；
- 指代模糊或信息省略：多轮对话中依赖上下文的代词或简略表达（如“它怎么样？”）缺乏显式语义；
- 复合意图交织：单条查询包含多个目标（如“检索+分析”），需解耦为原子任务；
- 口语化与冗余噪声：填充词、非结构化表达降低语义密度；
- 关键上下文缺失：未明确时间范围、业务单元、数据粒度等必要约束条件。

查询重写是将用户原始输入的自然语言查询，基于上下文、领域知识、对话历史及下游任务需求，转化为语义更明确、结构更规范、信息更完备且适配目标系统的标准查询表示的过程，从而提升后续下游任务的效能。

5.2.2 什么时候需要查询重写

查询重写是智能体构建中的关键技术，其核心目标在于对用户原始输入进行语义增强与结构优化，以适配下游任务的处理需求。最典型的场景包括：意图识别前的查询重写、RAG 检索前的查询重写。

1. 意图识别前的查询重写

在对话系统中，用户原始查询往往存在表述模糊、省略上下文或术语不规范等问题，直接影响意图分类模型的判别性能。为此，需结合对话历史、用户画像及预定义的意图标签体系，将原始查询转化为结构清晰、意图显式、术语规范的中间表示。该过程不仅有助于消除歧义、补全隐含语义，还能显著提升意图识别模块的准确率与稳定性，确保用户真实诉求被精准归类。

2. RAG 检索前的查询重写

在基于检索增强的生成框架中，原始查询若直接用于向量或关键词检索，常因信息不完整、术语不一致或语义分散而导致召回质量下降。此时，需依托领域术语词典、实体知识库及目标知识库的 Schema 结构，对查询进行语义聚焦与信息补全，重构为信息完备、术语统一、语义明确的检索友好形式。此类重写能有效提升检索阶段的覆盖率与相关性，为后续生成提供高质量上下文支撑。

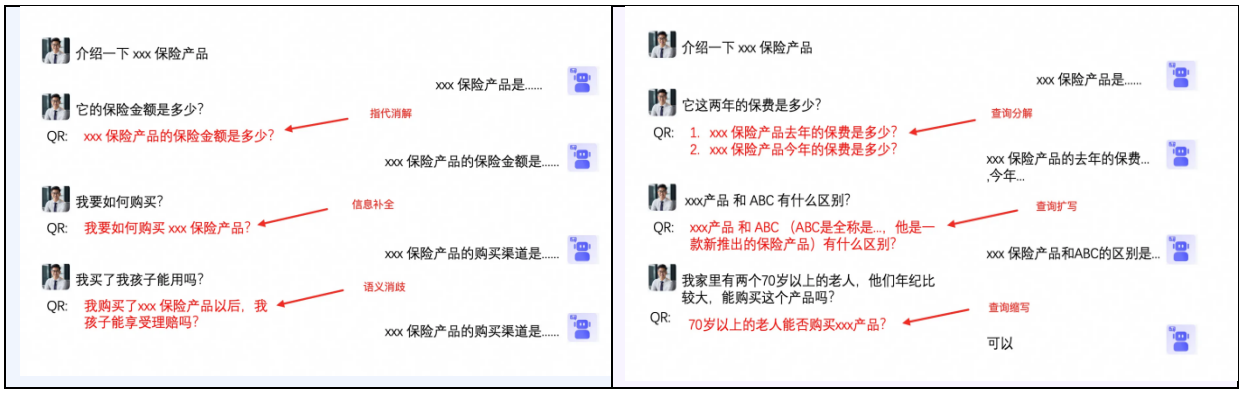
除上述典型场景外，查询重写在智能体架构的多个关键环节中亦发挥着重要作用。例如，在工具调用前，通过对原始查询进行重写，可生成符合工具接口规范的结构化指令；在任务规划阶段，重写后的查询可为后续规划模块提供清晰、结构化的任务描述与上下文信息。此外，查询重写还可用于多轮对话状态管理、跨模态对齐等场景，显著提升智能体整体的语义理解与任务执行能力。

5.2.3 优秀的查询重写应具备哪些核心能力

查询重写的关键能力通常涵盖如下：

1. 指代消解：结合对话或会话历史，识别并替换指示代词，还原其所指的具体实体或概念。
2. 信息补全：基于上下文推断用户意图，补充查询中缺失但语义必需的显式主体、属性或约束条件，以形成完整、可执行的查询。
3. 语义消歧：利用上下文线索及领域知识，对多义词、同形异义词或模糊表述进行精准语义解析，确保查询意图的唯一性与准确性。
4. 查询分解：依据规则库、语义模板或大语言模型的能力，将复杂查询拆解为多个逻辑清晰、可独立处理的子查询，以支持后续并行或串行执行策略。
5. 查询扩写：借助领域专家词典、同义词库、知识图谱或嵌入表示，引入相关术语、近义表达或上下位概念，提升召回率与语义覆盖度。
6. 查询缩写：剔除停用词、冗余修饰语及无关噪声，保留核心语义单元，在保证意图不变的前提下实现查询精炼，提升处理效率与匹配精度。

为更直观地理解上述查询重写能力，可见下图 Case 示例：



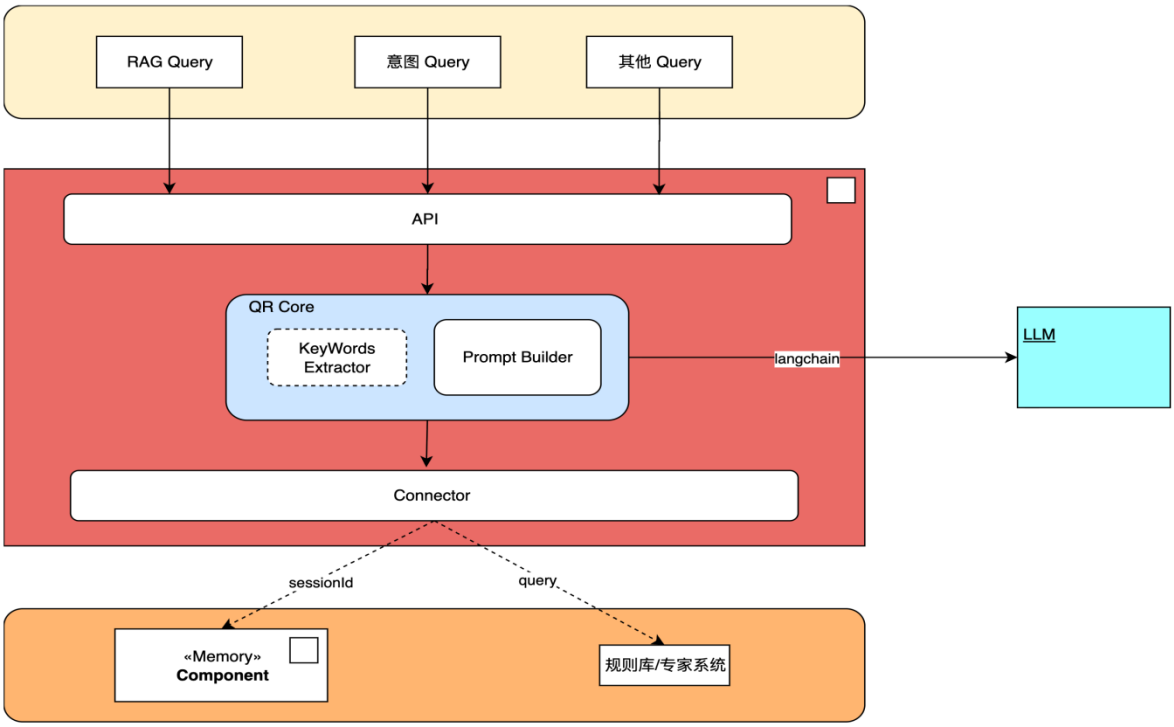
5.2.4 查询重写的主流技术实现方式

根据技术实现方式，主要有以下三类：

1. 基于规则模板的查询重写：依赖预定义的规则或模板（如正则表达式、槽位填充模板），对特定句式或关键词进行匹配和转换。适用于高确定性、结构固定的业务场景，例如工单查询（“查一下昨天的订单”→“查询创建时间为昨天的工单”）。
2. 基于大模型提示工程的查询重写：利用大语言模型，通过精心设计的提示词引导模型生成重写后的查询。适用于需要快速上线、支持多轮对话或复杂语义理解的场景。
3. 基于微调小模型的查询重写：在通用预训练语言模型基础上，使用领域内标注数据进行微调。适用于需兼顾泛化能力与性能的复杂场景。

5.2.5 案例参考：基于大模型提示工程的查询重写

基于大模型提示工程的查询重写，一种简单架构如下所示。考虑 RT 和性能，架构上只考虑调用 1 次大模型得到最终改写后的查询。



- API 接入层：负责对外提供 API。
- QR Core：KeyWords Extractor 负责提取原始查询中的关键词信息；Prompt Builder 负责将从其他组件中获取到的 context 与 内置 prompt 组合，得到最终调用模型前的 Prompt。
- Connector：组件连接器，负责连接和调用其他依赖的组件。

Prompt 示例如下：

你是 xxx 搜索的查询优化器。你的任务是将用户的自然语言查询改写为更有效的自然语言查询。

给定一个用户的搜索查询，你必须执行以下操作：
1. 识别核心概念和意图
2. 结合 context，替换其中的指示代词
3. 结合 context，添加相关的同义词和相关术语
4. 结合 context，补充原始查询中缺少的主体
5. 移除无关的填充词
6. 结构化查询以突出关键术语
7. 在适用的情况下加入技术或领域特定术语
8. 仅提供优化后的搜索查询，不要包含任何解释、问候或附加评论。

示例输入：
"xxx"
示例 context：
"xxxxxx"
示例输出：
"xxxxxxxxxxxxx"

约束条件：
1. 仅输出增强后的搜索词
2. 保持对可搜索概念的聚焦
3. 包含具体和通用的相关术语
4. 保留原始查询中的所有重要含义

原始 Query : {$(query)}
Context : {$(contex)}
改写后的 Query :

6 提示词工程：高效指令设计与系统化管理

在准确理解用户需求之后，如何引导大模型生成契合业务目标的响应，已成为 AI 落地的关键挑战。提示词已不再仅仅是简单的输入文本，而是承载业务意图、约束模型行为、决定输出质量的核心工程资产。高质量的 AI 应用不仅依赖于强大的基础模型，更取决于能否以系统化的方式对提示词进行设计、管理与持续优化。本章聚焦“高效指令设计与复用机制”，系统阐述提示词的核心构成要素、关键技术方法及企业级工程实践路径，强调将提示词从临时性脚本升级为可维护、可复用、可评估的标准化配置单元。通过构建融合角色设定、任务指令、上下文注入、示例引导与输出约束的结构化提示框架，并配套版本控制、自动化测试与迭代优化机制，企业才能有效保障 AI 输出在准确性、一致性与业务适配性上的稳定表现，真正将提示词工程融入可靠 AI 系统建设的核心流程。

6.1 提示词与提示词工程

你是否以为“写个问题丢给大模型”就是提示词？实际上，一个未经设计的提示，可能浪费 90%的模型潜力。

本节将帮你厘清：什么是真正的提示词？为什么它已从“对话技巧”演变为“工程资产”？提示词工程又如何成为企业 AI 落地的隐形基础设施？带着这三个问题，重新认识你每天都在使用的“输入框”。

6.1.1 什么是提示词

提示词（Prompt），是用户向大语言模型（LLM）发出的、用以引导其执行特定任务并生成相应回应的指令或输入。它是一门融合了语言学、计算机科学、认知心理学和特定领域专业知识的交叉学科，核心是通过精心设计、迭代优化与模型进行交互的指令序列（即提示词）来引导、约束和激发模型的潜在能力，使其输出的内容在准确性、可靠性、安全性和价值性上达到最优。提示词的质量，直接定义了 AI 应用能力的上限。

以下是一组提示词示例：

“请模仿 xx 的文风，写一篇关于当代年轻人在平凡生活中寻找诗意与温暖的短篇散文，要求语言质朴细腻、富有生活气息，字数约 500 字。”

“你是一位智能数据科学小助手，专注于电子商务领域，特别是‘xx’业务，具有丰富的数据科学与数据挖掘知识。
请根据我提供的用户问题：‘{QUESTION}’，结合如下知识内容进行回答。
{KNOWLEDGE}
你的任务是：1. 对用户输入的‘xx’进行同义词扩展，理解其等同于‘xx’业务。2. 推荐不超过四张最相关的表。3. 推荐列表必须按照提供的知识中记录的‘权重’值降序排列。4. 对于每一张推荐的表，必须提供其表名、表说明文档、以及从其关联的高频 SQL 逻辑中提炼出的 1-2 个常用统计 SQL 示例。5. 最终输出请使用 Markdown 表格格式。”

6.1.2 什么是提示词工程

提示工程是一门系统性地设计、优化和管理输入提示词的学科，目标是引导大型语言模型产生准确、相关、连贯且有用的输出。它涵盖了从选择恰当的词语、构建清晰的句子结构，到提供必要的上下文和示例等一系列技术和方法。这门学科正迅速从一种个人技巧演变为企业构建可靠 AI 应用不可或缺的核心工程能力。

“提示词”是静态的文本产物，而“提示词工程”则是一个动态的、贯穿始终的生命周期，它包括起草、测试、评估和持续优化的循环，直到输出质量达到预定标准，这种迭代的、系统化的特性是两者最核心的区别。

提示工程是连接业务需求与 AI 模型能力之间的关键桥梁。LLM 本身是一个通用的、经过海量数据训练的工具，但它并不知道用户具体的业务场景和目标，通过精心设计的提示词，企业可以将业务意图精确地传达给模型，引导其在特定场景下（如客户服务、市场营销）表现得像一个领域专家。一个好的提示词能够显著提升 AI 输出的质量和相关性，减少错误和“幻觉”，从而直接提高 AI 应用的实际效果与业务价值。

6.2 怎样写出高质量提示：提示词技术

为什么同样的模型，有人输出精准如专家，有人却得到一堆废话？差距不在模型，而在提示词的设计技术。

本节将系统拆解高质量提示的五大核心要素，介绍角色设定、思维链、少样本示例等关键技术，并提供可立即上手的编写原则。读完后，你将能像工程师一样“编码”你的意图，而非靠运气碰出好结果。

6.2.1 一个好提示由哪些关键部分组成

一个生产级、高性能的提示词通常由以下五个结构化组件构成：

1. 角色/身份

为模型分配一个明确的角色，引导 AI 调用该角色背后的知识体系、语言风格和思维方式，提供更专业、更符合场景的答复，避免通用、模糊的答案。

示例：“假设你是一名专业的皮肤科医生，为我们的健康科普公众号写一篇关于夏季科学防晒的文章”。

(1) 指令/任务

指令，明确、直接、无歧义地告知模型需要执行的具体任务，它是提示词最核心的部分，是模型行为的直接驱动力。任务描述必须具体、可操作，避免歧义，使用明确的动词（如：撰写、总结、翻译、列出、生成等），将宏观的目标分解为可执行的动作。

示例：“第一步，列出我们智能手表产品的三大核心卖点，第二步，为每个卖点撰写一段 50 字以内的描述”。

(2) 上下文

上下文信息提供模型完成任务所必需的背景信息、相关数据、目标受众等相关信息。为 AI 提供决策依据，使其输出内容更贴合您的具体业务、产品、品牌调性或受众需求。

示例：“背景：我们是一家面向初创企业的 SaaS 公司（产品是 XX 项目管理软件），目标受众是 20-30 人的技术团队负责人。写一份营销邮件，突出我们新发布的‘团队协作看板’功能，这个功能的说明在附件中。”

(3) 示例

根据是否在提示中提供示例，基础的提示技术可分为三种核心类型，即零样本提示(Zero-shot Prompting)、单样本提示(One-shot Prompting)、少样本提示(Few-shot Prompting)。这些技术都基于大型语言模型的“情境学习”（In-Context Learning, ICL）能力，即模型能从提示中直接给出的示例中学习并模仿，而无需重新训练。

(4) 输出格式/约束

明确定义输出的的期望类型、格式、结构、长度、语言或风格。确保模型回答不仅在内容上“正确”，更在形式上“可用”的关键环节。

示例：

- 格式要求：“请以 JSON 格式返回结果，根对象必须包含‘id’ (string), ‘name’ (string), 和 ‘tags’ (array of strings)三个键。”
- 结构要求：“你的回答必须包含三个部分，分别以‘## 摘要’、‘## 优点’和‘## 缺点’作为标题。”
- 风格要求：“回答的语气应保持专业、客观、中立，避免使用任何感性的、主观的或推测性的词语。

6.2.2 主流提示技术有哪些：零样本、少样本与思维链

6.2.2.1 零样本提示(Zero-shot Prompting)

零样本提示是最直接的提示形式，即只向模型提供任务指令，不包含任何示例。模型完全依赖其在预训练阶段获得的知识来理解和执行任务。其关键优势在简单、快速、成本低。

适用场景：适用于简单、通用的任务，例如基础的文本分类、简单问答或内容摘要，这些任务在模型的训练数据中已经有大量存在。

示例：

##请为以下内容生成专业摘要。

##摘要需严格遵循以下要求：

1. **核心聚焦**：仅保留原文的核心主题、3 个关键论点和 1 个重要结论
2. **信息完整性**：不得添加任何原文未提及的信息或个人观点
3. **客观中立**：避免使用主观形容词（如"非常"、"惊人"），仅陈述事实
4. **字数控制**：严格控制在 120-150 字之间（当前字数：[自动计数]）
5. **结构化输出**：必须包含"主题：[主题名称]，关键点：[1. ... 2. ... 3. ...]，结论：[结论]"

##内容：
[在此粘贴需要摘要的长文本]

6.2.2.2 单样本与少样本提示(One-shot & Few-shot Prompting)

当零样本提示效果不佳时，可以在提示中提供一个（单样本）或多个（少样本）完整的输入-输出示例，以引导模型的行为。这种技术被称为“情境学习”（In-Context Learning），模型能从提示中直接给出的示例中学习并模仿，而无需重新训练。

适用场景：当任务需要遵循特定的格式、风格或复杂的模式时，少样本提示非常有效。例如，进行情感分析、代码生成或从非结构化文本中提取特定实体。通过提供样本能有效引导输出结构，同时保持提示词简短；显著提高复杂任务的准确性和一致性。

示例：

##请分析以下客户评价的情感倾向，输出格式为：情感类型（积极/消极/中性），理由：[简要分析]

##示例：
评价："这款手机电池续航太棒了，用了整整两天才充一次电！"
情感类型：积极，理由：使用了"太棒了"等积极形容词，描述了产品优势

##现在请分析：
评价："产品包装破损，而且发货速度很慢，真的很失望。"

##请分析以下客户评价的情感倾向，输出格式为：情感类型（积极/消极/中性），理由：[简要分析]

##示例如下：
示例 1：
评价："这款手机电池续航太棒了，用了整整两天才充一次电！"
情感类型：积极，理由：使用了"太棒了"等积极形容词，描述了产品优势

示例 2：
评价："产品包装破损，而且发货速度很慢，真的很失望。"
情感类型：消极，理由：使用了"破损"、"很慢"、"失望"等负面词汇

示例 3：

评价："产品功能一般，价格有点高。"
情感类型：中性，理由：使用了中性描述"一般"，没有强烈正面或负面词汇

示例 4：
评价："物流很快，但产品质量不如预期，需要改进。"
情感类型：消极，理由：包含"但"转折词，前半句积极但后半句负面，整体倾向消极

##现在请分析：
评价："客服响应很快，但产品设计有点过时，希望下次更新。"

6.2.2.3 思维链提示

当任务需要复杂推理时，CoT 提示要求模型显式展示思考过程，引导模型模拟人类思维过程，将抽象问题拆解为可执行的中间步骤，通过逐步分解问题来提高输出的可解释性和准确性。该技术尤其适用于逻辑推理、多步骤决策等场景。

示例：

请解答以下问题，并严格按照以下步骤进行：
问题：一个长方形的周长是 36 米，长是宽的 2 倍。求这个长方形的面积。

请按以下步骤思考并作答：
设未知数
列方程：根据公式建立等式
解方程：求结果具体数值
给出最终答案：以完整句子形式呈现结果，包括单位

6.3.1 写提示不是碰运气：提示词核心编写原则

6.3.1.1 表述清晰具体

这是提示词工程的第一法则，必须尽量使用精确、量化、无歧义的语言，避免使用“一些”、“大概”、“更好”、“分析一下”等含糊其词的词汇。研究表明，在大多数情况下，更长、更具体的提示词通常会比短而泛的提示词产生更高质量的输出。

示例：

市场战略分析系统指令
角色：您是[企业名称]的高级市场战略顾问，负责为[目标市场]制定进入策略。请**严格使用 Tree of Thoughts (ToT)框架**进行分析，并**必须按以下格式输出**，不得添加任何额外内容：

思维树构建要求

1. 创建 3 个独立策略分支（每个分支代表一种市场进入模式）
2. 为每个分支包含 3 个评估维度：
   - 市场潜力（1-10 分，基于行业报告）
   - 风险评估（1-10 分，政治/汇率/文化/合规风险矩阵）
   - 收益预测（ROI%，3 年财务模型估算）

量化评估标准（必须使用）
维度	评分标准	量化方法
市场潜力	1=极小, 10=极大	IDC/Gartner 报告数据
风险评估	1=极低风险, 10=极高风险	四维风险矩阵加权
收益预测	ROI 百分比	3 年现金流贴现模型

强制输出格式（必须 100%遵循）
1.....

请帮我制定一个进入东南亚市场的策略，考虑各种因素。

6.3.1.2 结构化

当提示词包含多个逻辑部分（如系统指令、上下文信息、用户问题、输出示例等）时，使用清晰、明确的分隔符（如三重引号"""、三重反引号```、XML 标签<tag></tag>、或简单的###）来划分不同的区域，可以带来两大好处：

- 结构清晰：帮助模型更好地理解提示词的层次结构，避免将指令误认为上下文，或将用户输入误认为示例。
- 提升安全：在一定程度上可以抵御“提示词注入”攻击，即用户试图通过输入恶意指令来覆盖或篡改你的原始系统指令。

示例：

系统指令
你是一位资深供应链战略顾问，负责为[企业名称]优化全球供应链网络。请使用 Tree of Thoughts (ToT) 框架进行深度分析，构建思维树并评估不同优化方案。

上下文信息
- 企业名称：[企业名称]
- 业务类型：[例如：全球快消品制造企业]
- 当前供应链痛点：[例如：物流成本占营收 18%，交货周期平均 45 天]
- 优化目标：[例如：在保持服务水平的前提下，将供应链成本降低 15%]
- 关键约束：[例如：不改变现有工厂布局，需在 12 个月内完成]

分析要求
1. 创建 3 个独立优化策略分支（每个分支代表一种供应链优化模式）
2. 为每个分支包含 3 个评估维度：
  - 成本节约潜力（1-10 分，基于行业基准）
  - 实施可行性（1-10 分，技术/组织/时间约束）
  - 服务水平影响（1-10 分，客户满意度/交付准时率）

输出格式
1.....

你是一位资深供应链战略顾问，负责为[企业名称]优化全球供应链网络。请使用 Tree of Thoughts (ToT) 框架进行深度分析，构建思维树并评估不同优化方案。创建 3 个独立优化策略分支（每个分支代表一种供应链优化模式），为每个分支包含 3 个评估维度：成本节约潜力（1-10 分，基于行业基准）、实施可行性（1-10 分，技术/组织/时间约束）、服务水平影响（1-10 分，客户满意度/交付准时率）。企业名称：[企业名称]，业务类型：[例如：全球快消品制造企业]，当前供应链痛点：[例如：物流成本占营收 18%，交货周期平均 45 天]，优化目标：[例如：在保持服务水平的前提下，将供应链成本降低 15%]，关键约束：[例如：不改变现有工厂布局，需在 12 个月内完成]。输出格式：# 供应链优化方案评估：[企业名称]全球供应链网络 ## 优化策略分支 1：[策略名称] - **关键行动**： 1. [具体行动] 2. [具体行动] - **评估结果**： - 成本节约潜力：[分数]/10 - 实施可行性：[分数]/10 - 服务水平影响：[分数]/10 - **关键发现**：[1-2 句总结] ## 优化策略分支 2：[策略名称] ...（其他部分类似）客户查询：请为[企业名称]制定全球供应链优化方案，目标是在保持服务水平的前提下将供应链成本降低 15%。

6.3.1.3 增加约束

对于企业级应用而言，输出的可靠性和安全性至关重要。通过在提示词中加入明确的约束条件，可以有效地为模型的行为划定“护栏”，规避不希望出现的输出内容。约束可以分为两类：

- 正向约束（必须做什么）：`你的回答必须包含...`、`最终结果必须四舍五入到小数点后两位。`
- 负向约束（绝不能做什么）：`绝对不要...`、`避免使用...`、`禁止包含任何...`

示例：

##你是一个专业的财务分析助手。你的任务是根据下面提供的[季度财务报告]文本，生成一份面向管理层的业绩摘要。

##你的输出必须严格遵循以下要求：
必须只使用 [季度财务报告] 中提供的信息，不得引入任何外部知识或数据。

##摘要必须包含以下三个部分，并使用指定的标题：
1. 关键财务指标:
2. 业务亮点

3. 报告中提及的风险与挑战

####在关键财务指标部分，必须列出并总结报告中的“总收入”、“净利润”和“毛利率”这三个指标。所有引用的财务数据必须注明货币单位（例如：美元、人民币）。

##最终输出必须以 Markdown 格式呈现。

##[季度财务报告]:
“{此处粘贴财报文本}”

帮我看看这份财报，总结一下重点。

{此处粘贴财报文本}

6.4 企业如何规模化管理提示词：构建可持续演进的提示词工程体系

当你的团队有 100 个提示词在不同业务线运行，如何确保它们不退化、不冲突、可复用？临时复制粘贴的提示词，在规模化 AI 应用中注定失败。

本节将带你构建企业级提示词工程体系：从开发、测试、版本控制到持续评估，把提示词从“个人脚本”升级为“标准化配置单元”。如果你正考虑将 AI 嵌入核心业务流程，这一节就是你的操作蓝图。

6.4.1 从开发到上线：提示词的全生命周期怎么管

将提示词硬编码于应用程序代码中会导致系统紧耦合，不仅阻碍敏捷迭代，还限制非技术角色（如产品经理、领域专家）参与优化过程。推荐将提示词作为独立的可配置资产进行管理，实现与应用逻辑的解耦。由此，提示词的更新无需重新部署服务，显著缩短迭代周期，并支持跨职能协作。

提示词开发应遵循结构化的工程流程，其生命周期通常包括以下六个阶段：

- **需求分析**
    - 明确任务目标、输出形式、预期评估指标
    - 获取相关的数据集
- **初始提示词构建**
    - 按照标准结构编写初始提示
    - 可参考历史相似业务场景提示词或已有模板
- **效果测试与评估**
    - 使用客户数据集进行测试

- 计算关键评估指标（如 precision 等）

- **提示词优化**
  - 根据评估结果反馈进行人工分析及优化迭代
  - 根据评估结果反馈使用 APE 进行优化迭代
- **版本管理与部署**
  - 将优化后的提示词纳入版本管理
  - 在线上服务中部署并监控效果
- **运维和迭代**
  - 根据线上反馈效果进行持续优化迭代
  - 在模型升级或场景扩充时进行提示词适配

上述流程的有效执行，依赖于标准化的提示词配置单元与配套的测试评估机制，详见后续章节。

6.5.1 把提示变成可复用的“配置单元”

6.5.1.1 一个标准提示配置单元包含什么

为支撑提示词的工程化管理，企业需定义最小可管理单元：提示词配置。该配置完整描述了一次 LLM 调用所需的全部上下文与参数，是版本控制、测试与部署的基本对象。

将提示词配置作为最小单元进行管理的意义在于：

- **确保可复现**：同样的提示词文本在不同模型版本或不同 `temperature` 设置下，会产生截然不同的结果。将三者绑定管理，能确保测试和生产环境中的行为一致，并能够精确地复现任何历史行为。
- **可对比测试：**这种整体性视角是进行 A/B 测试的基础，它允许团队隔离变量：例如，使用相同的“提示词模板”和“模型参数”，仅测试不同“模型”的影响；或者固定“模型”和“参数”，来比较两个不同“提示词模板”的优劣。
- **可审查：**在金融、医疗等受监管行业，当 AI 系统做出一个关键决策时，必须能够精确追溯是哪个提示词、哪个模型版本、以及在何种参数下生成的该结果。对完整的“提示词配置”进行版本控制，可支持精细化的审查。

一个完整的提示词配置至少包含以下三个核心组成部分：

1. **提示词模板**：定义 LLM 输入的结构化文本，包含：
   A. 静态内容：如角色设定、任务指令、输出格式要求等；

B. 可变内容：也称为“占位符”或“变量”，是在运行时动态填充的数据，这些变量通常用花括号（如 {user_query} 或 {customer.name}）表示。可变内容可以来源于：

- 用户输入（如聊天消息）
- 外部系统数据（如从数据库或 API 获取的订单信息）
- 对话历史上下文
- ……

2. **目标模型：明确指定所使用的模型及其具体版本（例如 qwen-plus-2025-01-25），模型的微小更新都可能导致输出行为的巨大变化，固定模型版本是保障输出一致性的关键。**

3. **模型参数：一个高性能的提示词配置，不仅包括文本内容，还包括与之协同工作的模型参数，这两者共同决定了最终的输出结果。关键参数如：**

A. Temperature：控制输出的随机性与创造性，取值范围通常为 [0.0, 2.0]。值越低，输出越确定；值越高，多样性越强。

- 低温度 (如 0.0 - 0.3)：输出更具确定性、可预测性。模型会倾向于选择概率最高的词。适用于需要事实准确性的任务，如代码生成、问答和报告总结。
- 高温度 (如 0.8+)：输出更具多样性、创造性。模型会增加选择低概率词的可能性。适用于头脑风暴、创意写作或生成多种营销文案的场景。

B. Top-p：这个参数通过限制候选词的范围来控制输出的多样性。它会选择一个最小的词汇集合，使得这个集合中所有词的累积概率大于或等于设定的 p 值，然后模型只从这个集合中进行采样。在实践中，Top-p 通常被认为比 Top-k（选择概率最高的 k 个词）更优，因为它能根据上下文动态调整候选集的大小。

- 低 Top-p (如 0.1)：候选词汇非常有限，输出更可预测，随机性较低。
- 高 Top-p (如 0.95)：候选词汇范围广，输出更具创造性，随机性较高。

示例：

```
{
 "prompt_id": "CUST_SUPPORT_TICKET_SUMMARIZER_V1.2",
 "version": "1.2",
 "author": "jane.doe@example.com",
 "description": "Summarizes a customer support ticket into key points for escalation.",
 "components": {
  "role": "You are an expert technical support analyst.",
  "context": "{ticket_history}", // 动态注入的变量
  "task": "Summarize the following support ticket conversation. Identify the core customer
```

```
issue, the steps already taken, and the current status.",
   "style": "Use a bulleted list.",
   "tone": "Concise and professional.",
   "audience": "Tier 2 support engineers.",
   "constraints": {
    "output_format": "json",
    "json_schema": {
     "issue": "string",
     "steps_taken": ["step1", "step2"],
     "status": "string"
    },
    "max_length_words": 150,
    "banned_terms": ["guarantee", "promise"]
   }
  },
  "model_parameters": {
   "model": "gpt-4-turbo",
   "temperature": 0.2,
   "top_p": 1.0
  }
}
```

6.5.1.2 模板化使用：如何避免重复造轮子

提示词模板是将业务逻辑与大语言模型调用解耦的关键机制。其核心是通过结构化方式动态生成上下文相关的提示词代替硬编码的静态文本。常见使用模式按复杂度递进如下：

- **变量注入：**应用程序在调用 LLM 之前，会将实时数据“注入”到模板的占位符中，生成一个完整的、针对当前情境的提示词。适用于简单场景（如 3-5 个变量）。

```
# 企业级参数
company_name = "ABC 消费品"
market = "东南亚"
current_resources = "$5M, 50 人技术团队"
strategic_goal = "3 年内获取 15%市场份额"

# 基础变量注入模板
prompt_template = (
   "你是一位资深市场战略顾问，负责为{company_name}制定进入{market}的市场进入策略。"
   "当前资源：{current_resources}。战略目标：{strategic_goal}。"
   "请使用 Tree of Thoughts 框架分析，创建 3 个策略分支并评估维度。"
)

# 实际数据注入
```

```
prompt = prompt_template.format(
    company_name=company_name,
    market=market,
    current_resources=current_resources,
    strategic_goal=strategic_goal
)

print(prompt)
```

- **使用模板引擎：**为了处理复杂的逻辑（如循环、条件判断），建议使用成熟的模板引擎，这比手动的字符串格式化更强大、更安全。它允许开发者创建包含变量占位符和逻辑结构的模板文件，然后在运行时将实际数据动态填充到模板中，生成最终的文本内容。

```
from jinja2 import Template

# 企业级参数
context = {
    "company_name": "XYZ 集团",
    "market": "欧洲",
    "current_resources": {"funding": "$10M", "team": 100},
    "strategic_goal": "2 年内获取 20%市场份额",
    "show_resources": True  # 条件开关
}

# Jinja2 模板（包含条件逻辑）
template = Template("""
### 系统指令
你是一位资深市场战略顾问，负责为{{ company_name }}制定进入{{ market }}的市场进入策略。

{% if show_resources %}
### 当前资源
- 资金储备: {{ current_resources.funding }}
- 技术团队: {{ current_resources.team }}人
{% endif %}

### 分析要求
1. 创建 3 个独立策略分支
2. 为每个分支包含 3 个评估维度：
  - 市场潜力（1-10 分）
  - 风险评估（1-10 分）
  - ROI 预测（3 年）
""")
```

```
# 生成提示词
prompt = template.render(**context)

print(prompt)
```

- **集成开发框架：**像 LangChain 这样的开发框架提供了强大的 `PromptTemplate` 和 `ChatPromptTemplate` 类，可以方便地定义、管理和组合带有变量的模板，并与 LLM 调用无缝集成。

```
from langchain.prompts import PromptTemplate

# 定义 PromptTemplate（带变量和格式）
prompt_template = PromptTemplate(
    template="""
### 系统指令
你是一位资深市场战略顾问，负责为{company_name}制定进入{market}的市场进入策略。

### 上下文
- 目标市场: {market}
- 当前资源: {current_resources}
- 战略目标: {strategic_goal}

### 分析要求
1. 创建 3 个独立策略分支
2. 为每个分支包含 3 个评估维度：
   - 市场潜力（1-10 分）
   - 风险评估（1-10 分）
   - ROI 预测（3 年）
""",
    input_variables=["company_name", "market", "current_resources", "strategic_goal"]
)

# 企业级参数
context = {
    "company_name": "GlobalTech",
    "market": "中东",
    "current_resources": "$8M, 75 人团队",
    "strategic_goal": "18 个月内获取 12%市场份额"
}

# 生成提示词（自动填充变量）
```

```
prompt = prompt_template.format(**context)

# 与 LLM 集成（示例：调用 OpenAI）
from langchain_openai import ChatOpenAI
llm = ChatOpenAI(model="gpt-4-turbo", temperature=0)

response = llm.invoke(prompt)
print(response.content)
```

6.5.1.3 版本控制让每次提示词修改可追踪

所有“提示词配置”都应存储在版本控制系统中。它为所有配置提供了一个单一、可追溯的事实来源，记录了每一次变更的历史。

为了系统地传达提示词配置变更的类型和影响，企业应采用语义化版本控制思想对提示词配置进行版本标记：

- 主版本（MAJOR）:用于不向后兼容的、破坏性的变更。例如，修改了输出 JSON 的 Schema，导致下游解析逻辑中断。
- 次版本（MINOR）:用于向后兼容的功能增强。例如，向输出 JSON 中增加一个新的、可选的字段。
- 修订版（PATCH）:用于进行向后兼容的问题修复。例如，修正提示词中的拼写错误或调整措辞以提高清晰度。

6.6.1 如何验证提示是否真的有效：测试与评估框架

6.6.1.1 评估框架选择

系统化的评估框架是提示词质量保障的核心基础设施。依赖人工主观经验的调优方式，已经无法满足企业级 AI 应用对提示词可靠性、稳定性的要求。评估框架不仅需要量化提示词的功能效果，还需要覆盖效率（token 消耗、响应延迟）、鲁棒性（抗干扰）、安全性（合规拦截）等多维指标。以下是几个主流框架：

- **Langfuse：**Langfuse 是一个面向 LLM 应用的开源可观测性和分析平台。在评估方面，Langfuse 支持设置“LLM-as-a-judge”评估和人工标注流程，以比较不同模型、Prompt 和配置的表现。平台还内置数据集管理功能，方便创建测试集和基准数据，以进行持续的预部署测试与分组实验。
- **Promptfoo：**Promptfoo 是专为 LLM 应用设计的一站式自动化测试工具。它允许开发者通过简单的 YAML 配置或命令行快速定义测试用例，并支持多模型并行评测。Promptfoo 拥有丰富的断言体系，用户可以进行精确字符串匹配、包含检查、正则匹配或模型进行评估。开发者可以通过插件接入自定义模型或断言逻辑，并可集成至 CI/CD 工作流中。

- **OpenAI Evals：**OpenAI Evals 是 OpenAI 官方开源的评估框架。它通过 JSONL 数据集和 YAML 配置定义评测任务，支持基础的匹配、包含和模糊匹配等评估方式。评测输出以准确率为核心指标，同时给出置信区间等统计信息。

6.6.1.2 离线测试与在线测试配合

离线测试：

- 单元测试：基于预构建的测试用例集（含典型场景、边界条件及历史缺陷案例），评估提示词在给定输入下是否生成符合预期的输出。
- 回归测试：在每次变更后，运行整个测试套件来监控性能，如果回归测试失败，则不做部署。

在线测试：

A/B 测试是在真实用户流量中验证提示词配置优劣的黄金标准，实施需遵循以下原则：

- 明确假设：制定清晰、可衡量的假设。
- 控制变量：隔离单一变量（如仅改变提示词文本，保持模型和参数不变）。
- 随机分流：将用户流量均匀分配至对照组与实验组。
- 多维监控：追踪关键业务指标（转化率、满意度）、用户体验（响应时间、交互轮次）及系统成本（Token 消耗、延迟）等指标。
- 统计验证：确保实验周期足够长、样本量满足统计显著性要求。

6.6.1.3 能否让模型自己参与评估

对于缺乏客观标准的主观任务（如摘要质量、创意文案、情感倾向），不存在唯一的“正确答案”，此时，可使用一个功能强大的 LLM（“裁判模型”）来对应用 LLM 的输出进行打分。其实施应遵循以下流程：

- 与领域专家共同定义评估标准。
- 创建人工标注的基准数据集。
- 精心设计裁判提示词，包含评估标准和示例。
- 校准与验证，确保裁判模型的评分与人类专家的一致性。

7.知识增强：RAG 与实时信息获取

知识增强是现代智能体突破大模型固有知识边界、实现动态认知与事实对齐的关键能力，其核心在于通过外部信息的实时注入，弥补模型参数化知识的静态性、滞后性与领域局限性。在真实应用场景中，仅依赖预训练模型内部知识的智能体极易产生“幻觉”、提供过时信息或无法处理专业领域问题。知识增强通过 RAG 工程与联网搜索两大技术路径，构建起智能体与外部知识世界的动态连接，使其具备“知之为知之，不知则查之”的类人认知机制，从而显著提升回答的准确性、时效性与专业性。

7.1 用检索增强生成（RAG）让回答更准、更专、更可靠

你是否担心大模型“一本正经地胡说八道”？当它面对内部没有学过的专业知识、最新数据或敏感业务文档时，该如何避免幻觉并给出可靠回答？别急。RAG（检索增强生成）正是解决这一困境的利器。

本节将带你从基础 RAG 起步，逐步探索图增强、智能体驱动乃至企业级安全管控的进阶方案，并通过一个简单案例展示结合 RAG 技术、具备知识安全管控能力的检索智能体。读完本节，你将掌握让 AI“不懂就查、查得准、用得安”的核心能力。

7.1.1 什么是 RAG？为什么你的智能体需要它

传统大语言模型受限于其训练数据的静态性，有很多局限：知识时效性受限（知识截止于训练完成时）、对专业领域及长尾知识覆盖不足，以及易产生“幻觉”（即生成看似合理但事实错误的内容）。

RAG 通过将外部知识检索与文本生成相结合，构建了一种动态融合权威信息的架构范式。其核心理念是在生成响应前，实时从可信、可更新的知识源中检索与用户查询高度相关的上下文，并以此引导模型生成更准确、可靠的回答。

RAG 不仅是缓解模型幻觉、增强领域适应性的关键技术，更是构建智能体的核心支撑：它赋予智能体按需访问最新、权威或私有信息的能力，相当于为其配备“外挂记忆”。这种机制不仅支持个性化、可解释的决策与交互，还通过解耦知识存储与模型推理，大幅提升了系统的灵活性、可维护性与持续演进能力。

RAG 的核心流程由三个紧密结合的阶段构成：

1. 索引（Indexing）：对原始文档（如企业手册、技术资料、客户记录等）进行预处理，包括文本分块、嵌入向量化、元数据标注及索引构建，形成结构化的可检索知识库。索引质量直接决定后续检索的覆盖广度与精度。

2. 检索（Retrieval）：用户发起查询时，系统将其映射为与索引空间一致的向量，并从知识库中召回最相关的文本片段。高阶实现常融合关键词匹配、多路召回或重排序策略，以提升结果的相关性与鲁棒性。

3. 生成（Generation）：将检索所得上下文与原始查询共同构造提示（prompt），输入生成模型产出最终回答。该阶段关键在于精准融合外部信息，确保输出内容准确、连贯、紧扣任务目标，同时规避无关或矛盾内容。

7.1.2RAG 能解决哪些实际问题

RAG 被广泛用于构建知识驱动的问答系统，典型应用包括智能客服、企业内部知识助手、销售与技术支持等场景：

- 智能客服场景：利用 RAG 技术构建能实时响应客户咨询的自动化服务系统。通过理解自然语言提问，从产品手册、常见问题库（FAQ）、历史工单等结构化或非结构化数据中精准检索相关信息，为用户提供个性化的解答，显著提升服务效率和客户满意度。
- 企业内部知识助手：利用 RAG 可帮助员工快速获取公司制度、项目文档、技术规范或过往经验等内部知识资源，通过自然语言交互高效定位所需信息，降低知识获取门槛，提升组织协同效率。
- 销售与技术支持场景：销售人员可通过 RAG 驱动的工具即时查询产品参数、竞品对比、定价策略等资料，辅助客户沟通；技术支持人员则能基于用户描述的问题，自动检索故障排查指南、维修记录或技术白皮书，快速生成专业、一致的解决方案，缩短响应时间并提高服务质量。

除此之外，RAG 不仅作为信息检索与生成的桥梁，更在智能体构建的多个核心环节中提供关键支撑，显著增强智能体的感知、推理与决策能力。具体而言：

- 规划环节：RAG 支持动态知识注入，使智能体在任务分解与路径规划过程中能够实时检索外部知识库中的上下文相关信息。例如，将思维链提示、业务规则、操作流程或历史成功案例作为可检索的知识片段，在规划时进行查询与融合，从而生成更合理、可执行且符合领域规范的行动计划。
- 记忆环节：RAG 有效扩展了智能体的长期记忆能力。通过将结构化或非结构化的经验数据（如过往交互日志、用户偏好、任务结果等）存储于向量数据库或知识图谱中，智能体可在后续交互中按需检索相关记忆片段，实现个性化响应与持续学习。
- 查询重写环节：在面对模糊、简略或术语混杂的用户输入时，RAG 可检索领域术语表、实体定义、同义词映射或常见问题模板，辅助智能体重构查询语义，提升后续检索与理解的准确性。例如，在医疗或法律等专业领域，系统可基于检索到的专业词汇解释自动将口语化表述转化为标准化术语，从而提高问答质量。
- 工具选择调用环节：RAG 可为智能体提供关于可用工具、API 接口文档、调用示例及约束条件的实时参考信息，辅助其判断何时调用何种工具，并生成符合规范的调用参数。

RAG 通过将外部知识动态融入智能体的认知闭环，在规划、记忆、理解、决策等多个维度实现知识增强，不仅提升了智能体的任务完成能力，也增强了其在复杂、动态和专业场景下的适应性与稳定性。

7.1.3 从基础到进阶：主流 RAG 方案怎么选

如前所述，RAG 在智能体及其构建过程中具有多样化的应用场景。基于 RAG 架构的演进与技术特性，企业可根据具体场景选用相应能力模块，从而构建适配度最优的 RAG 解决方案。下表按场景分类，列示了典型方案及其实施的关键要素，可快速参考，也可在后续子章节详细查阅每种方案的特点。

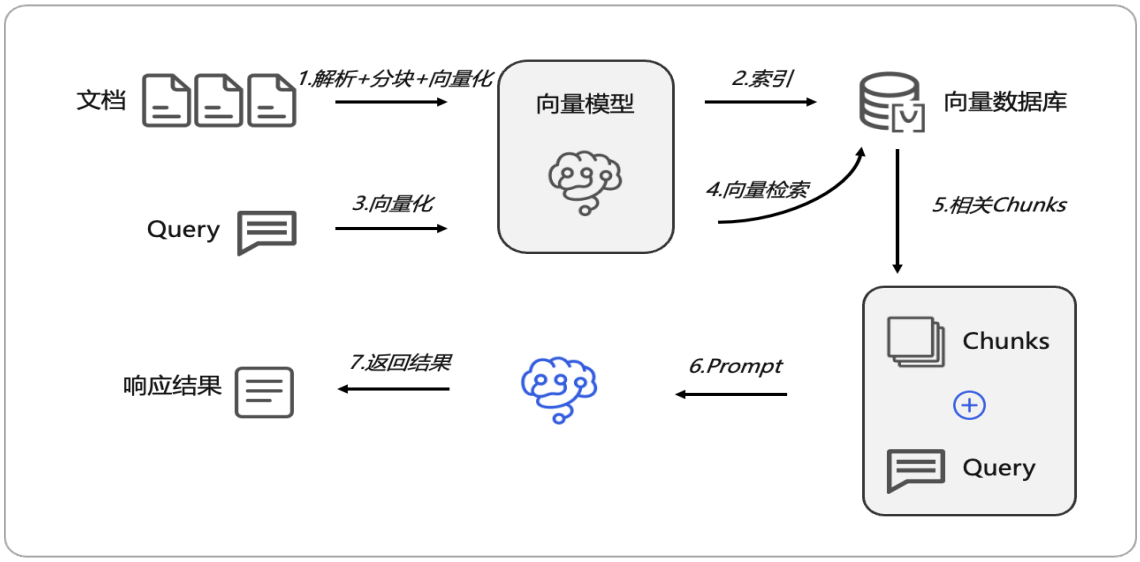
	简单 RAG 方案	增强 RAG 方案	图增强 RAG 方案	Agentic RAG 方案	企业级 RAG 方案

方案定位	快速响应简单查询	提升复杂/模糊问题的召回准确率与完整性	跨实体多跳推理	自主推理、任务规划与动态执行	多源知识统一治理与安全管控
适用场景	适用于封闭域内单轮问答与知识检索的轻量级应用及快速原型验证。	面向专业领域中高准确率要求、支持模糊与不完整查询的语义理解与精准问答。	支撑多跳推理与复杂关系挖掘，适用于需深度关联分析的高阶认知任务。	驱动自主任务分解与深度研究的智能代理，胜任复杂分析与决策辅助场景。	构建跨业务、合规可控的企业级知识中枢与智能服务底座。
关键技术特征	● 单一向量检索 ● 无查询改写 ● 无结果重排序 ● 上下文直接拼接	● 索引增强 ● 查询优化 ● 混合检索（全文+向量） ● 重排序	● 实体关系三元组抽取 ● 图结构知识库构建 ● 图检索 ● 多跳推理	● LLM 驱动的 Agent 架构 ● 记忆与状态追踪 ● 工具调用（API/搜索/知识库） ● 迭代检索与结果验证	● 多源异构接入 ● 多模态支持 ● 知识算子和知识管道自定义 ● 高频问答库前置 ● 意图识别和查询路由 ● 多源融合检索 ● 终端用户细粒度权限控制
关键流程	提问 → 向量检索 Top-k → 拼接上下文 → 生成	查询优化 → 混合检索 → 重排序 → 生成	查询优化 → 全文/向量/图谱混合检索 → 重排序 → 联合生成	任务理解 → 子任务规划 → 动态检索/调用工具 → 迭代推理 → 输出	前置权限拦截 → 意图识别 → 检索路由(问答库/文档知识库) → 多源融合检索 → 后置权限过滤 → 安全生成
知识表示	文本片段	分块文本 + 关键词索引	文本 + 图谱（实体-关系三元组）	动态构建内部知识图谱与任务状态	多模态结构化数据 + 图谱 + 问答库
检索前处理	原始 Query 直查	查询多轮改写和增强	查询多轮改写和增强	自主问题拆解与子任务生成	意图识别 + 路由决策 + 查询重写
检索方式	单一向量检索	Hybird：全文检索+向量检索	Hybird：全文检索+向量检索+图检索	动态调用多种检索与外部工具（含 RAG、API、Web Search）	多源融合检索（本地+第三方 RAG 引擎）

检索后处理	无	语义重排序、结果融合	语义重排序、结果融合、多跳推理	信息交叉验证 + 一致性判断	多源结果融合
是否支持工具调用	否	否	否	是（可调用 API、搜索引擎等）	否
检索精度	一般	较高	高	高（动态优化）	高
响应速度	快	较快	较快	较慢	可调优
成本开销	低（少量 LLM 调用）	中等（增加查询优化与 Rerank）	中等（图谱构建前期投入大）	高（多步推理+多次调用）	高（多模块资源消耗）

7.1.3.1 简单 RAG 方案

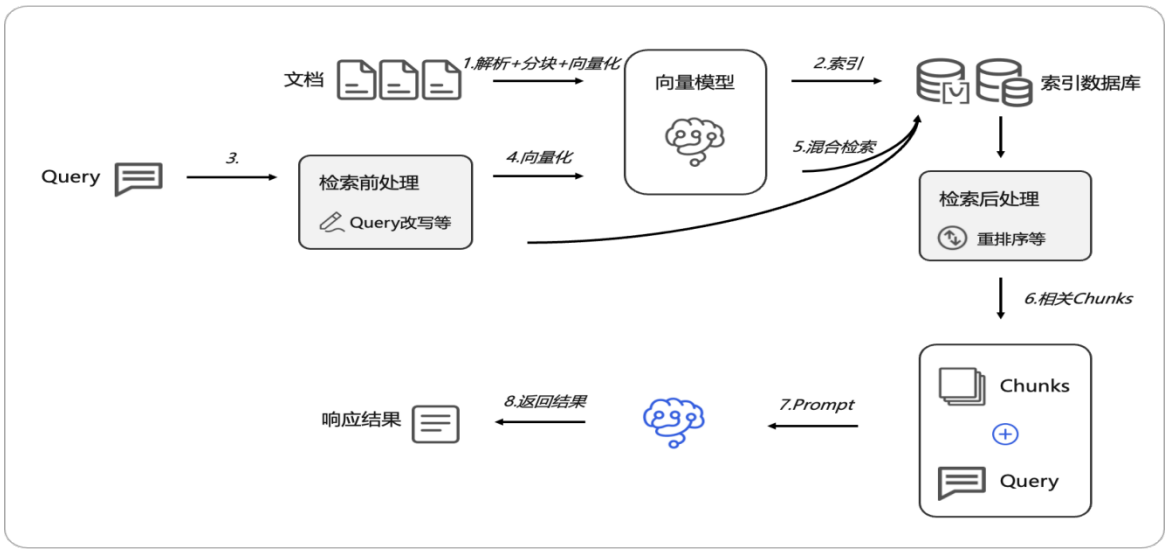
快速构建一个轻量、高频的问答系统，问题类型固定且集中，语义明确，知识结构清晰，关注响应效率与实现简易性，对生成结果的精度容忍度较高，无需复杂对话理解或多轮推理能力。



- 方案特点：采用“用户提问 → 向量检索 Top-k 片段 → 上下文拼接 → 大模型生成”的线性处理流程，基于语义向量索引实现知识召回与生成的直接串联。未引入查询优化与结果重排序机制，整体链路简洁高效。
- 优势：实现简单，链路简洁、响应速度快，适合高频但标准化知识服务的快速上线。
- 局限：检索性能高度依赖原始查询的表达质量，难以应对模糊、多义或信息缺失的提问；缺乏上下文理解与多轮对话建模能力；不支持复杂语义推理或多跳信息整合。
- 适用场景：封闭域、单轮问答场景：如企业内部 HR 政策查询、IT 支持自助服务、产品手册检索等。快速试点项目或概念验证阶段（PoC/MVP）。

7.1.3.2 增强 RAG 方案

对回答准确性、信息完整性及语义理解深度有较高要求，常见于专业领域决策支持场景。原始查询可能存在指代不清、语义模糊或信息缺失问题，仅靠基础检索难以满足高质量输出需求。希望系统能主动优化查询表达、融合多源信息并提升上下文相关性排序能力。



- 方案特点：在简单 RAG 的基础上引入检索前处理、混合检索和检索后处理环节，形成“查询优化 → 混合召回 → 结果重排序 → 增强生成”的闭环流程，显著提升检索相关性与答案准确性。

  A. 检索前处理：通过引入 Query 组件，在检索前对原始查询进行多轮对话改写（包括指代消解、信息补全与语义消歧）与 Query 增强（包括查询分解、查询扩写、查询缩写）。

  B. 混合检索：采用混合检索策略，融合全文检索与向量语义检索，兼顾关键词精确匹配与深层语义相似度，提升召回覆盖率与相关性。

  C. 检索后处理：引入结果重排序 Rerank，基于语义相关性对多源召回结果统一打分排序。

- 优势：显著提升复杂语义问题的召回率与精确率；支持模糊查询的理解与补全；可融合多源信息，提升最终生成内容的完整性与准确性。

- 局限：链路复杂度上升，推理延迟略有增加；需额外算力支持查询改写与重排序模块。

- 适用场景：对回答准确性要求较高的专业领域问答系统；存在大量模糊、口语化或不完整提问的知识服务场景；如：法律条文解释、医疗指南查询、金融产品说明等。

7.1.3.3 图增强 RAG 方案

面临涉及多个实体间复杂关系的问题，传统基于文档片段的 RAG 容易因“信息孤岛”导致推理断裂。例如“某高管曾在哪些关联公司任职？”、“该药物适用于哪些由特定基因突变引起的疾病？”等问题需要跨文档、跨段落的逻辑串联与关系推导，仅靠局部语义匹配无法有效解决。

- 方案特点：在增强型 RAG 的基础上，引入图结构知识表示与多跳关系推理能力，突破传统基于文本片段匹配的“信息孤岛”瓶颈，实现跨文档、跨段落的逻辑关联与路径推导。具体来说：

  A. 实体关系抽取：定义领域关键实体类型（如人名、组织、疾病、药物等）；从非结构化文本中自动抽取出三元组（头实体-关系-尾实体），如“爱因斯坦 - 导师 - 韦伯”。

  B. 图索引构建：基于抽取的三元组构建图知识库；图谱与向量索引并行构建，形成多路检索底座。

  C. 多跳推理：基于图遍历算法的路径查询，实现“一跳→多跳”的关系推理。

  D. 联合检索机制：同时激活全文检索、向量检索与图检索，输出融合结果；图检索结果可用于增强生成上下文的逻辑连贯性与可解释性。

- 优势：实现从“局部匹配”到“全局推理”的跃迁；提升复杂关系类问题的回答准确率与推理透明度；支持知识间的语义连接与因果推导。

- 局限：图谱构建依赖高质量的信息抽取模型与领域实体预定义。

- 适用场景：适用于强关系建模与复杂推理任务，典型场景如：

  A. 医疗诊断辅助：症状→疾病→基因→药物的链条推理。

  B. 法律案件分析：当事人→合同→法条→判例的关联挖掘。

  C. 金融风控：企业→股东→担保→关联交易的穿透式核查。

  D. 科研知识发现：学者→论文→机构→合作网络的探索。

7.1.3.4 Agentic RAG 方案

在该方案下，系统需要具备类人类的思维链、自主规划与工具调用能力，以应对开放域、复杂任务，而传统 RAG 被动响应查询，无法处理需多步推理、外部验证或动态调整策略的问题。

- 方案特点：方案将 RAG 升级为 Retrieval Agent（检索智能体），嵌入自主推理与任务执行工作流中，实现从“被动响应”到“主动探索”的范式转变。其核心特征如下：

  A. 基于 LLM 的推理引擎驱动：采用 ReAct（Reasoning + Acting） 或 Plan-and-Execute 范式，实现“思考→行动→观察→反思”的闭环。

  B. 任务分解与计划生成：将复杂问题拆解为子任务序列，例如“查找 A 公司 AI 产品线 → 分析 B 公司专利分布 → 比较技术路线差异 → 综合判断趋势”。

  C. 动态检索与迭代优化：每一步调用 RAG 模块获取信息，结合上下文判断是否需要进一步检索或调用其他工具（如检索领域知识库、调用领域 API、联网搜索）。

  D. 记忆与状态追踪：维护中间推理状态与历史上下文，避免重复检索，提升效率与一致性。

E. 验证与反馈机制：支持交叉验证多个来源信息，识别矛盾点并发起二次检索，确保最终输出的可靠性。

1. 优势：具备类人类的思维链与自主决策能力；可完成开放域、非结构化、跨模态的研究型任务；输出更具可解释性与逻辑严谨性。

2. 局限：运行成本高（LLM 调用频次多、Token 消耗大）；响应延迟较长，需精细设计 Prompt 模板与工具接口；对 Agent 框架稳定性与错误恢复机制要求高。

3. 适用场景：高阶认知任务与自动化研究场景，典型场景如：咨询公司行业研究报告自动生成，投行并购项目中的标的公司尽调分析，企业战略部市场趋势预测与竞争格局分析，科研机构文献综述与分析生成助手。

7.1.3.5 企业级 RAG 方案

本方案面向企业级 AI 知识中枢建设需求，针对多源异构知识资产整合、复杂权限体系治理及多业务系统集成等核心挑战，构建统一、安全、可扩展的知识全生命周期管理平台。

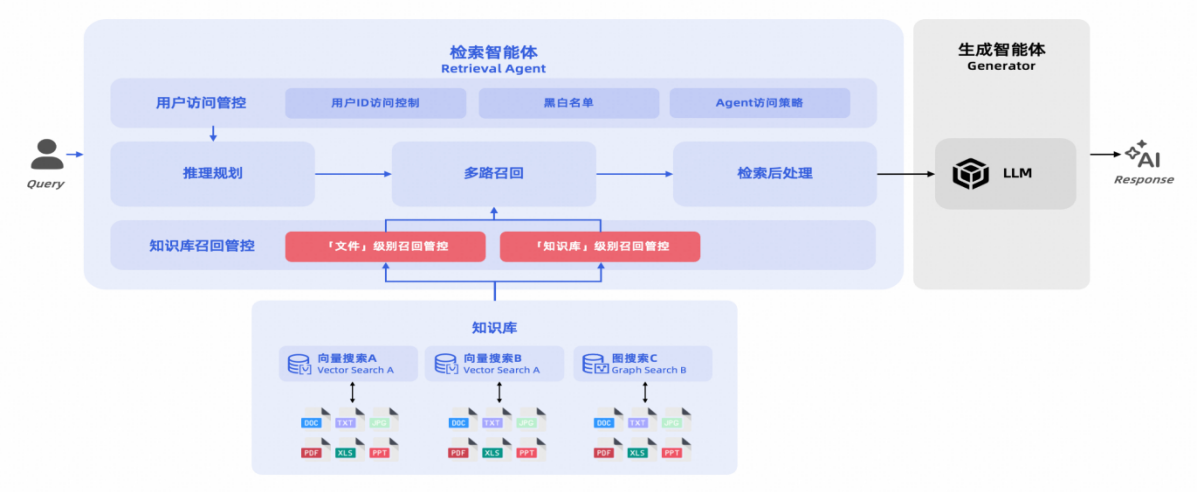
方案在图增强多跳推理 RAG 架构基础上，深度强化知识准备、知识检索与知识安全管控三大企业级能力。在知识准备层，支持文档、网页、外部协作平台等多源异构数据接入，结合多模态解析与语义增强技术，并提供可编排的专业数据算子与自定义知识处理管道；在知识检索层，通过前置高频问答库实现毫秒级响应，显著降低大模型调用频次与 Token 开销，同时融合意图识别驱动的智能查询路由机制，支持本地知识库与第三方 RAG 引擎并行检索与结果精排融合；在安全管控层，构建覆盖知识库级、文档级乃至段落级的分层细粒度权限体系，实现端到端的访问控制与动态权限过滤。

该方案适用于企业级 AI 知识中枢、跨业务线智能客服/数字员工底座，以及金融、医疗等对合规性与安全性要求严苛的行业场景，为多业务线高效复用高质量知识资产提供坚实支撑。

7.1.3.6 案例参考：一个具备知识安全管控的检索智能体

在企业场景中，知识库需强化知识安全管控。下图展示了典型“检索智能体”架构通过双重机制保障安全：

- 分层权限体系：实现从知识库级到文档级的细粒度访问控制。
- 端到端权限过滤：在检索阶段即依据用户身份、角色、黑白名单及 Agent 策略，动态过滤无权限内容，确保输出结果不泄露敏感信息。



7.2 联网搜索：让智能体随时获取最新信息

如果你的智能体只能依赖训练截止前的知识，它如何回答“今天某股票价格是多少？”或“最新出台的 AI 监管政策有哪些？”这类问题？答案是：赋予它“上网查资料”的能力！

本节将揭秘如何通过集成联网搜索（如阿里云信息查询服务），让智能体突破静态知识的牢笼，实时获取权威、动态的外部信息。

7.2.1 为什么联网搜索是智能体的“实时知识外挂”

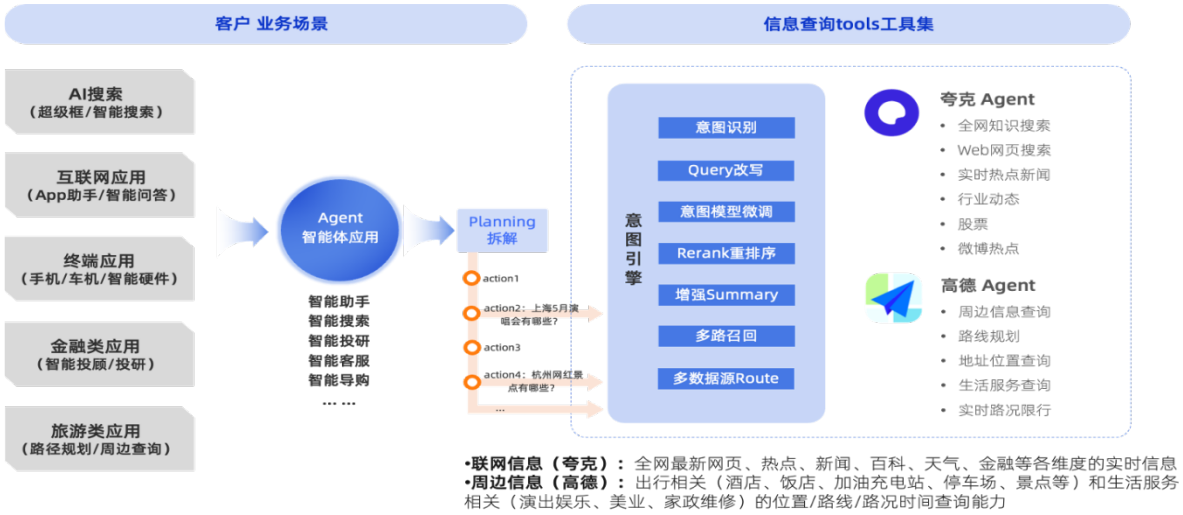
联网搜索是智能体通过实时接入互联网，动态获取开放域、高时效性信息的核心能力。其运作机制为任务驱动、语义引导与结果验证的闭环流程：首先基于用户意图进行深度理解与查询重构，生成高精度搜索关键词；随后调用搜索引擎 API 获取候选信息源；继而通过可信度评估、多源交叉验证与关键事实抽取，对信息进行结构化处理；最终将可溯源、可验证的“活知识”融入响应输出。

该能力对智能体的重要性体现在以下四方面：

- 突破信息时效瓶颈：弥补大模型训练数据截止带来的滞后性，支持对近期事件、实时数据（如股价、天气）、最新产品等动态信息的准确响应。
- 抑制事实性幻觉：通过外部权威信源替代内部参数化记忆，显著降低在精确数据、专业领域或人物履历时的虚构风险。
- 实现信源可追溯：提供透明的信息出处，满足企业级应用对可信度、合规性及深度研究的需求，增强用户对回答的信任与可控性。
- 拓展认知深度与广度：针对复杂或小众问题，整合全网多元信息，生成超越模型内生知识的全面、深入见解。

7.2.2 案例参考：通过阿里云信息查询服务，实现智能体的权威信息实时查询

阿里云信息查询服务为大语言模型提供先进的、多源融合的信息服务层。通过整合夸克通用搜索引擎的公共互联网信息、自建垂类高质量信息以及专业领域信息（如高德地图），为大模型提供一个动态、精准、可信的外部信息源。该服务有效弥补通用 LLM 在实时性、信息广度及外部知识依赖方面的不足，同时可应用于基于外部数据的模型训练增强。



服务核心特点如下：

- 多源信息融合：突破单一数据源的局限。覆盖全网海量公开信息，满足模型对新闻时事、生活常识、热点趋势以及周边服务查询等泛在知识的需求。
- 查询意图精准识别：深度理解用户的真实意图，智能改写和拆解。
- 请求结果精准: 精简、提炼返回的海量原始数据，保留原始信息结构的智能切片和语义排序。为 LLM 提供精简、去噪、有序的信息源，提升其生成回答的准确性和效率。

8.专属模型定制：定制化后训练

在通用大模型能力日益成熟的背景下，尽管通用大模型展现出强大的泛化能力，但其在企业特定业务场景的知识深度、场景有关的上下文理解、以及面对复杂流程性指令的遵循方面仍存在固有局限。因此，如何构建具备企业专属能力的 AI 应用，其关键已从模型的通用能力转向其在特定场景下的专业化深度。

本章节旨在阐述如何通过高效、可控的技术手段，将通用大模型打造为深度契合企业自身需求的专属模型。模型后训练作为核心解决策略，通过在高质量、领域特定的私有数据集上进行二次训练，来实现将通用知识向专业能力的精准迁移。

8.1 为何需要专属模型

基于通用基础模型，通过整合行业知识、企业私有数据及业务垂直场景的特定数据集，实施二次训练与参数精调。此举旨在构建一个具备“专业性强化、可控性提升、综合成本效益优化”的专属模型。

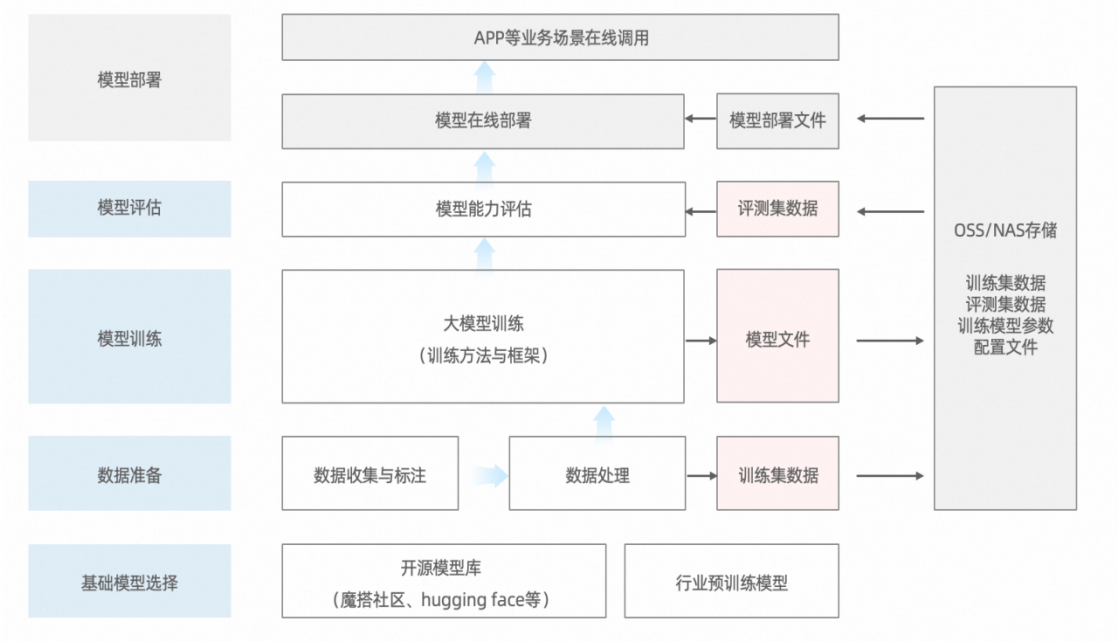
模型训练的必要性判断

当基础模型难以契合特定业务需求时，可考量定制化训练的必要性。判断依据如下：

- 业务领域深度
    - 若业务流程涉及海量专业知识和行业术语，且仅依赖提示工程（PE）或检索增强生成（RAG）等轻量级技术，仍无法使基础模型准确理解并处理实际业务语义的场景，如医疗诊断、金融风控、法律文书解析等高度垂直的专业领域。
- 业务场景复杂度与性能要求
    - 在业务场景结构复杂，且对模型推理的准确性（Accuracy）或实时性（Latency）有较高要求时，若基础模型的当前性能表现无法满足预期的业务指标。

综上所述，若满足上述任一条件，即基础模型难以契合特定的业务需求时，则需通过定制化训练专属模型来提供解决方案。

整体方案框架



8.2 典型应用场景

1. 行业大模型

- 场景描述
  - 在畜牧养殖行业，兽医知识体系复杂、实践门槛高且知识分布零散。在通用大模型基础上，引入兽医领域专业语料（专业书籍、临床病例、操作 SOP 等），对模型进行后训练。实现专业人员行业规范知识问答、养殖户日常防疫及辅助诊断。
- 适用行业
  - 同行业，及金融、法律等门槛较高，知识体系复杂的行业。

2. 2.交互助手分类模型

- 场景描述
  - 用户通过语音与智能硬件进行互动。传统方案存在复杂语境用户意图难识别、无法支持上下文理解等问题。基于高准确率，低响应时延，成本最优的业务要求，对小尺寸模型进行 SFT，提升识别准确率。
- 适用行业
  - 有语音助手、智能客服等场景的行业客户。

3. 多模态信息识别模型

- 场景描述

- 信息服务中，档案的数字化流程需要大量人工作业。传统 OCR 模型在信息录入，识别准确率和识别速度都难以达到要求。通过对 Qwen-VL 模型微调，显著提升识别准确度，极大程度取代人工录入和校验，降低人力成本，缩短流程周期。

- 适用行业
  - 医疗、制造、零售、安防等行业客户，用于识别特定物品、规范标准和内容理解的场景。

8.3 主流训练方法和框架选择

8.3.1 训练方法介绍

1. 继续预训练(CPT)

在预训练模型的基础上，加入海量无监督领域知识数据（TB 级别，数十亿 token）、并混合通用数据进行训练，从而提升在专业领域上的表现。能够深刻理解领域术语、行业“黑话”、逻辑关系，提升在该领域所有下游任务的性能上限。适用于知识密集型行业，如生物医药、法律、金融、科研等。

2. 监督微调(SFT)

- 全参微调
  - 更新所有参数，需要大量计算资源，在数据较少的情况下容易过拟合。
  - 适用于目标任务与预训练任务差异较大或需要大幅提升模型性能的场景。
- 高效微调
  - 通过调整少量关键参数或引入额外参数来实现模型适应新任务的目的。以 LoRA 为例，训练参数量仅 0.1%~1%，训练时间短，不会破坏大模型原有的知识。
  - 适用于任务相对接近通用语义的场景，训练数据量在几千到十几万指令对。

3. 强化学习(RL)

- PPO
  - 利用人类反馈训练奖励模型，再通过 RL 循环优化，使模型生成符合人类偏好的输出。
  - 通用性强，适用范围广，常用于复杂场景，如机器人控制、游戏等。
- GRPO
  - 通过分组采样和组内奖励归一化替代价值模型，提高复杂推理任务的训练效率。
  - 适用于大模型场景中的数学、代码类任务。
- DPO

- 直接优化模型偏好，跳过奖励模型训练，通过成对比较（如“哪个回答更好”）来调整策略参数。训练稳定，计算效率高。
- 适用于偏好数据充足的任务，如对话生成等。

8.3.2 训练框架介绍

1. LLaMA-Factory

简单易用且高效的大语言模型训练与微调平台。可以在无需编写任何代码的前提下，在本地完成上百种预训练模型的增量预训练、指令监督微调、RL 训练等，支持优化算法、加速算子、推理引擎和实验监控。

2. ms-swift

魔搭社区提供的模型训练部署框架，支持 500+大模型与 200+多模态大模型的训练（预训练、微调、RLHF 训练）、推理、评测、量化与部署。模型开发者可在 ms-swift 框架中一站式完成各类训练需求。

3. transformers/pytorch

最常用的深度学习框架，兼具便捷性和灵活性。通过与 Hugging Face 模型中心的无缝对接，能以极简代码调用海量模型；同时，与 PyTorch 的深度集成保留完全的底层控制力，支持开发者训练高度定制化的大模型。

8.4 训练前准备

你手上有业务数据，也选好了开源大模型，但为什么训练出来的效果依然不理想？问题很可能出在“准备阶段”。基座模型是否真的适合你的任务？私有数据是否覆盖了关键场景？标注质量是否经得起推敲？

本节将带你系统梳理训练前的三大关键决策：如何科学选型基础模型、高效构建领域数据集，甚至在数据稀缺时通过合成手段“无中生有”，为专属模型的成功训练铺平道路。

8.4.1 基础模型选型策略

1. 业务场景与模型规模的匹配评估

市面上主流的开源模型参数量通常分布在 0.5B 至 72B 的区间内。基础模型的选择应根据具体的业务需求与性能指标进行权衡：

业务场景	建议模型规模	选型依据
意图识别	7B/14B 或更小尺寸	此类任务常作为 AI 应用的前序环节，基于对低延迟（Low Latency）的要求，通常推荐选择小型参数量的模型。
工具/插件调用	14B 或更大尺寸	模型的选择需视待调用工具的数量和参数结构的复杂程度而定，一般建议选择 14B 左右的模型以保障效果。

特定风格文本生成	32B 或更大尺寸	对模型的语义理解及上下文信息处理能力要求较高，通常需要具备更大尺寸参数的模型来承载。
结构化信息提取/行业知识注入	32B 或更大尺寸	需根据任务的复杂度和所需的知识深度来决定，以确保提取和注入的准确性。

1. 基础模型核心能力评估

在确定了大致的规模范围后，需对候选基础模型的性能进行深度评估：

- 参考公开榜单：查阅业内权威的公开排行榜，获取模型在通用能力上的基准表现。
- 功能覆盖度：重点评估模型的关键技术指标，包括上下文窗口长度、对工具调用的支持情况，以及是否支持慢思考等复杂推理机制。
- 验证集评测：使用企业私有数据集对候选模型进行严格的效果评测，以验证模型的领域知识覆盖度和推理结果的质量。

2. 算力资源与成本效益评估

模型规模与计算资源之间需进行审慎的平衡：

- 模型规模与资源消耗：通常而言，参数量巨大的大型语言模型（例如 Qwen3-235B）虽然具备更强的推理能力和潜在知识深度，但其在训练和推理阶段对机器资源的需求量和时间成本也显著增加。
- 成本效益原则：必须在模型的预期性能与现有计算资源及预算之间找到最佳的平衡点，避免资源过度投入或性能无法满足业务目标。

8.4.2 训练数据收集策略

1. 数据收集与规模规划

训练数据的收集工作应根据所选基础模型的规模，准备相应体量的高质量标注数据。

- 规模验证：在训练初期，建议首先采用**小规模数据集（万级样本量）**进行快速迭代，以验证和评估模型训练的初步效果。
- 分布均衡：确保数据的分布具有全面性和均衡性，需尽可能覆盖所有的业务垂直场景，避免出现特定分类下的数据量过度集中或过于稀疏的极端情况，从而保证模型的泛化能力。

2. 常见数据来源

训练数据的来源应多元化，主要包括以下几个方面：

- 第三方公开数据：行业领域的专业书籍、学术文献、公开报告及行业标准等

- 企业内部私有数据：企业的规章制度、标准操作流程（SOP）、内部文档及历史知识库等。
- 线上生产数据：来自线上系统运行的日志、用户交互记录、工单信息等实际业务生产数据。

3. 数据标注规范

数据标注是确保模型训练质量的关键环节，可采用以下方式：

- 业务专家离线标注：由具备深厚业务经验的专家进行专业、高精度的离线标注。
- 在线标注平台：利用专业的标注工具或平台进行标准化、流程化的数据标注工作。

8.4.3 高质量训练数据合成策略

核心原则：在模型训练过程中，数据的质量（包括多样性、逻辑连贯性与复杂程度）比单纯追求数据量更为重要。

1. 基础数据操作

此阶段旨在对原始数据进行标准化和基础质量控制：

- **数据清洗**：运用预设规则及语义理解技术，对原始数据执行基础的脱敏处理、重复项移除（去重）和噪声过滤等操作。
- **问答对提取**：利用数据处理工具实现文件的上传、智能切分、问题创建，并匹配相应的答案生成。
- **问答对校正**：对已提取的问答对进行一致性、有效性检查，并依据业务需求进行精确校对与修正。
- **问答对合成**：结合校正后的问答对与业务真实数据，模拟生成符合实际业务场景的**“场景指令-回答”**结构化数据。

2. 数据增强操作

此阶段旨在提升数据的覆盖面和鲁棒性：

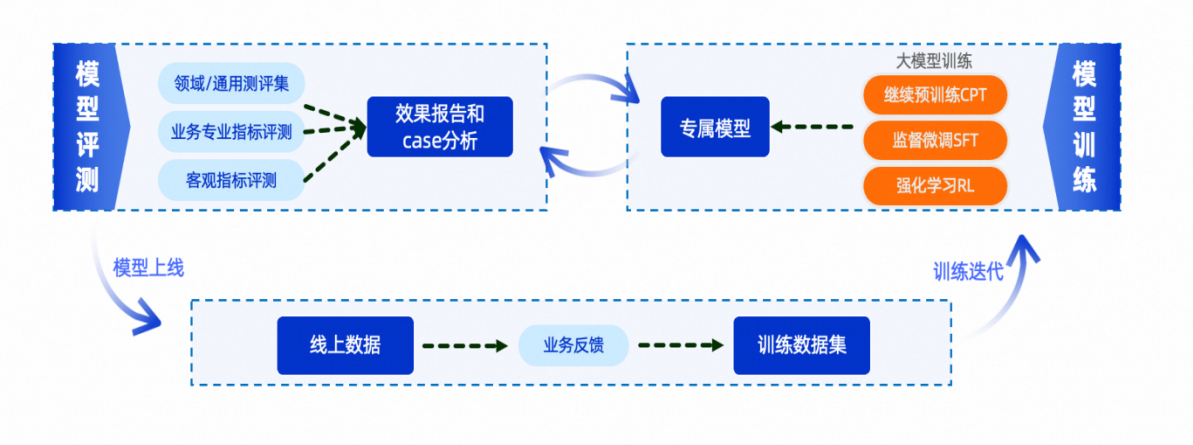
- **数据裂变**：在真实数据样本不足的情况下，可利用大型模型进行数据补充。常见方法包括：文本重组、语义改写，以及基于生成式的多样化增强等。
- **数据均衡**：针对不同分类数据量差异悬殊的情况，需进行分类平衡处理。低频数据采用数据模拟或对相同样本进行多次采纳等方式进行上采样；高频数据通过随机欠采样或清洗筛选出高质量样本的方式进行下采样。
- **鲁棒性处理**：采用噪声注入技术，通过模拟同音错别字、键盘位错别字、数字口语化等方式，增加训练数据的噪声，以提升模型在真实应用环境中的抗干扰能力。

- **思维链蒸馏**：将大尺寸模型所具备的复杂推理能力（即“思考过程”）通过特殊训练，转移和压缩到小尺寸模型中，以增强其推理表现。

8.5 训练与评估流程

模型开始训练了，然后呢？是盯着 loss 曲线干等，还是盲目相信最终指标？如何判断训练是否“学到了业务逻辑”而非死记硬背？又该如何设计评估体系，确保模型上线后不会在真实场景中“翻车”？

本节将为你拆解端到端的训练与评估闭环：从方法匹配、框架选择，到过程监控与多维效果验证，让你每一步都心中有数，避免“训完才发现方向错了”的昂贵代价。



8.5.1 确定训练方法

根据数据形态、算力资源规模及业务目标，可选用单一或组合式的模型训练方法。

1. 单一方法训练

	SFT	RL
训练数据	经过人工或机器标注的输入-输出样本对	人类偏好排序数据或效用评分数据集
主要作用	固定模型在特定格式任务上的表现，提升输出的确定性与准确性。	实现用户偏好对齐，同时提升模型泛化能力与安全性。
资源成本	适中（标注成本高，算力相对低）	较高（需在线采样标注）
风险	过拟合风险高，输出可能过于僵化或机械化（即“机械套用”）	训练过程稳定性挑战大，算力与数据开销大

2. 组合式训练

通过串联不同的训练方法，可以实现优势互补，以达到更复杂的模型目标：

- 持续预训练（CPT）→监督式微调（SFT）→强化学习（RL）
  - 首先通过 CPT 注入特定的领域知识语料，随后利用小规模标注数据执行 SFT 固化格式，最终通过 RL 完成用户偏好对齐。实现领域知识、输出格式与用户偏好的有效平衡
- 监督式微调（SFT）→强化学习（RL）
  - 先通过 SFT 建立高质量的基准回答，随后利用 RL 进行偏好对齐。在确保模型输出准确性的基础上，兼顾系统的稳定性和用户体验

8.5.2 选择训练框架

框架名称	优劣简述	选型建议
LLaMA-Factory	✅ 底层是 pytorch&transformers。 ✅ 有图形化界面，一键训推，无 coding 能力也可轻松上手。 ❌ 只支持常用的训练方法，参数调节、算法选择灵活性不如其他两个框架。	只做基本的 LLM/VLM 的 CPT/SFT/RL 训练，优先考虑此框架。
ms-swift	✅ 魔搭社区提供的官方框架，中文模型支持更全面，功能更全面，支持算法更多。 ✅ 支持命令行一键启动训练。 ❌ 安装环境时依赖包较多，有时会遇到依赖冲突。	非主流模型训练建议考虑使用此框架。
transformers/pytorch	✅ 几乎所有大模型训练都会用到的底层框架，卡型适配度高。 ✅ 理论上基于此框架可实现任何模型训推算法，具有广泛的社区支持和算法选择，非常灵活。 ❌ 使用需要算法技术背景，coding 比重大，模型训练加速优化、各种 AI-infra 实现需额外集成。	在 LLaMA-Factory 和 swift 都不支持的训练场景，且有一定算法技术背景，优先考虑此框架。

8.5.3 观察训练过程

- 过拟合
  - 现象：训练损失减小，验证损失增大。

- 导致结果：模型开始死记硬背训练集，对未训练过的数据表现降低，泛化性降低。
- 解决策略：
  - 收集更多数据，增加训练数据多样性和数据量。
  - 调整超参，如减少“训练步数”、降低“学习率”、仅使用部分参数训练、正则化损失函数等。

- 欠拟合（不常见）
  - 现象：训练损失没有明显变化或增大。
  - 导致结果：模型未学习到训练集的知识，即模型不收敛。
  - 解决策略：
    - 使用参数量更大的模型。
    - 检查数据质量，确保训练数据清洗充分，无噪音。
    - 调整超参，如增大“训练步数”、提升“学习率”、使用全参训练、选择合适的学习率 warm-up 策略、适当提升 batch_size 大小等。

8.5.4 评估训练效果

3. 评估维度

- 客观指标评估：
  - 多分类任务：指标包括 Precision、Recall、F1，综合考察模型推理的正确性与完整性。
  - 问答/文本生成任务：指标包括 BLEU、ROUGE，基于 n-gram 的重叠程度来评估模型生成文本与参考答案之间的语义相似度。
  - 语音合成任务：指标为 WER 词错误率，统计转写结果中发生的插入、删除、替换的词汇次数，衡量语音转文本的准确性。

- 业务自定义评估：
  - 裁判员模型评测：准备自定义评测集（包括指令-参考回答对），使用 Python 脚本自动化执行评测。通过大模型，自动对模型推理结果与参考答案进行打分或排序投票。
  - 人工评测：邀请业务专家对模型推理结果进行主观打分；对比不同模型的推理结果，进行排序对比。

- 领域/通用评估：

- 公开数据集评估：利用权威的公开基准数据集评估模型的通用能力和泛化性，常见包括 MMLU、C-Eval、GSM8K、CMMLU 等多语言、多学科或特定任务的综合性测试集。

4. 案例参考

客服系统需要处理用户通过文本（App、微信）提出的问题，将用户意图准确分类到预设的 40 多个意图类别。这是一个典型的多分类任务。通过对历史对话数据及分类打标数据，对 8B 模型进行精调，构建专属模型。

评估方案采用了客观指标评估+业务自定义评估结合的方式进行效果验证。

评估类型	评估指标	任务描述	评估结果
客观指标评估	Recall（召回率）	衡量真正属于某类别中，被模型正确识别出的比例。	90%+。大幅提升，漏判率降低，确保了所有有效意图都能被捕获。
	Precision（精确率）	衡量模型预测为某类别中，真正属于该类别的比例。	95%+。大幅提升，误判率显著降低，减少了将非意图 A 错判为意图 A 的情况。
	F1-Score（综合性能）	Precision 和 Recall 的调和平均数，综合衡量正确性与完整性。	90%+。相比基模，精调模型在客服场景下的综合意图识别能力达到了业务要求。
业务自定义评估	高频意图召回率	评估高频意图的召回率，不能漏判。业务专家复核 100 条样本的识别结果。	投诉召回率 100%，满足安全性要求。

9.数据工程：为智能体提供高质量数据燃料

在大模型发展过程中，若将模型算法视为驱动其思考与决策的核心引擎，那么数据工程便为其提供了高品质“数据燃料”、保障其稳定运行。其重要性不仅体现在模型训练的初始阶段——通过提供高质量、多样化的训练数据，从根本上提升模型能力的上限与可靠性的基准；更贯穿于智能体构建与迭代的全生命周期。从原始数据加工、标注，到通过检索增强生成（RAG）技术链接外部可信知识库以对抗“幻觉”，再到建立应用回流机制以驱动自我优化，每一个环节都依赖于一个成熟、高效且可扩展的数据工程体系。

9.1 数据工程介绍

在构建企业级大模型应用的过程中，普遍面临**知识边界受限、信息时效滞后、内容幻觉风险以及数据安全隐患**等挑战。上述痛点在垂直行业的特定应用场景中表现得尤为显著。为此，亟需引入行业专业知识与企业内部私有数据，以增强大模型的业务适配能力，确保生成内容的精准性与可靠性。

现阶段，企业内部数据资产主要由非结构化数据（如公开资料、原始文档、音视频文件）与结构化数据（如业务数据表、系统日志）构成。然而，大模型的落地应用高度依赖于规范化、标准化的数据集与知识库。因此，将企业原始多模态数据转化为适配大模型所需格式的过程至关重要，这一核心转化环节即「数据工程」。

下图展示了标准化的数据处理全流程：流程左侧为原始数据输入，右侧为大模型应用所需数据。数据工程贯穿其中，通过数据加工、增强、标注等操作，将原始数据转化为大模型可用的高质量数据资产。

9.2 企业数据的使用场景

1. 模型后训练

- 场景描述

- 面向行业/企业私域知识进行模型训练，弥补基础模型知识与能力边界；在同等算力下，高质量与多样化数据是提升模型能力的关键。

- 数据形态：
  - 预训练与领域语料：通识、行业专业知识。
  - 有监督数据：SFT 指令问答对、RLHF/DPO 偏好比较对。
  - 验证集数据：业务标准问答对。

2. 知识库构建

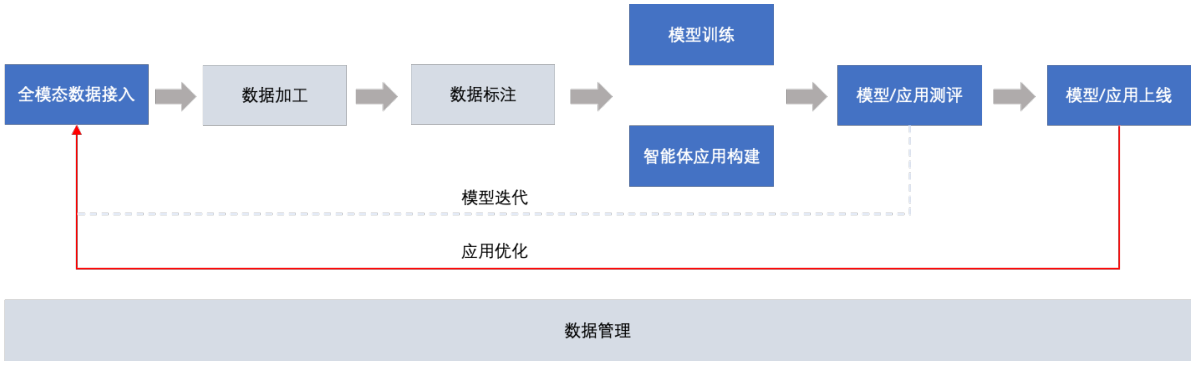
- 场景描述
  - 用于检索增强生成（RAG）、企业知识问答与决策支持，为模型提供可溯源的资料，降低“幻觉”。
- 数据形态：配套切片与嵌入索引以支持高质量检索。
  - 文档与半结构化资料：制度手册、FAQ、合同、工单/日志。
  - 结构化数据：业务表、指标库、知识图谱。
  - 多模态内容：图像/音频/视频。
  - 元数据：版本、来源、时间、权限、许可。

3. 线上应用优化

- 场景描述
  - 通过线上应用数据回流，持续优化智能体的提示词、工具调用策略与业务流程，提升应用准确率和稳定性。
- 数据形态：
  - 会话与交互日志：用户查询、模型回复、用户评价/采纳。
  - 工具/函数调用轨迹与结果：入参/出参、错误类型、可验证标签。

9.3 企业数据接入与评估

在进行数据处理之前，首先要接入业务数据，并进行基础评估，明确是否进入数据工程的链路中进行处理。



9.3.1 常见来源

数据来源	数据特点	示例
企业私域数据	● 格式丰富 ● 内容复杂 ● 文档内容可靠度高 ● 数据量较大	● 内部报表 ● 产品使用说明 ● 标准、规范文档 ● 日常操作记录
应用产出数据	● 贴合应用场景 ● 架构统一规范 ● 应用结果好坏未知 ● 会有不符合预期的数据	● 智能问答中的场景问答对 ● 流程化应用中，对流程的构建与应用结果
三方数据	● 可靠性和完整性未知	● 外部公开数据

9.3.2 接入数据评估

企业内部的数据来源多样且复杂，数据内容和质量差异比较大。错误或无效的数据会引发大模型结果的不可靠性，因此需要对数据进行初步评估。

- **数据的常见问题**
  - 原始文档的格式、内容，模型无法理解。
  - 文档内容中包含重复信息、无效信息。
  - 文档中存在敏感信息。
- **数据评估：针对这些数据常见问题，需要通过评估指标对接入的数据进行评估。**

- 文档数据常见评估指标：

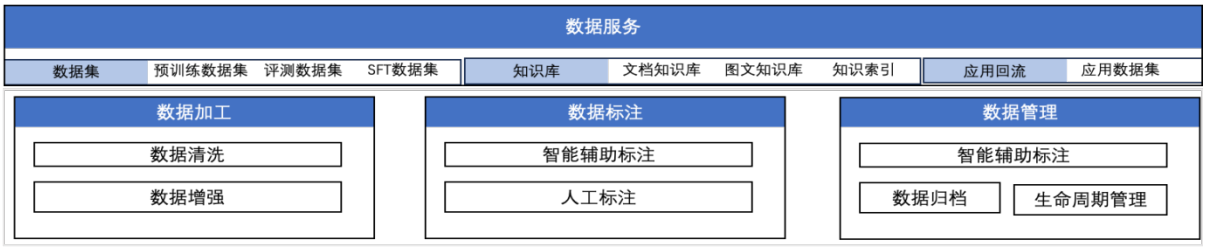
评估指标	说明
重复率	重复内容占总文档的比率，如 N-Gram
文档长度	文档字数，太短的文档可能没有实际意义
字母计数占比	字母或数字占文档的比例，占比过低说明文档中包含有用信息过少
困惑度	衡量模型预测下一个词的不确定度或困惑程度。困惑度越低，模型对数据的预测越准确
相似度	文本间的相似度，如 SimHash 海明距离
特殊字符占比	特殊字符占文档的比例
敏感词检测	是否含有黑名单中的敏感词
毒性检测	是否含有非法信息

- 多模态数据常见评估指标：

评估指标	说明
图像大小	高度/宽度过大的图像会在处理为模型输入后丢失大量的视觉信息
图像文本匹配度	图像文本匹配分数过低的图像可能是无效占位图像
视频时长	过滤过长或过短的视频，保证视频的质量
视频文本相似度	采样若干视频帧，计算视频帧画面描述和训练数据描述的文本相似度
视频文本区域占比	采样若干视频帧，计算视频帧的文本区域占比，过滤文本占比过大的数据

9.4 数据工程核心链路

数据工程的核心是如何为大模型提供高质量可靠的数据，核心链路包括：数据加工、数据标注、数据管理、数据服务四部分。



9.4.1 数据加工

数据加工的基础模块是算子，通过算子串联整个数据加工的 pipeline 流程，进行数据处理。例如对一个网络地址 url 进行处理，每一个节点（html 标签处理、email 处理等）都是一个算子。

根据算子类型不同，数据加工过程可分为数据清洗和数据增强。数据清洗主要去除脏数据，数据增强用于补充源数据。

1. 数据清洗

- **核心目的**
  - 去除原始数据中的干扰数据：去除乱码、重复、冲突等。
  - 对数据格式进行规范化：日期、数字规范表达等。
  - 对特定数据进行清理、加密：客户敏感信息脱敏，非法链接清理等。

- **主要方法**
  - 传统数据清洗模式，基于规则：

异常类型	说明	处理方式
非法字符	不可见字符、乱码	删除不可见、乱码字符
数据重复	上下文有重复	删除时间靠前的数据
数据冲突	上下文对同一事实描述不一致	删除时间靠前的数据
数据脱敏	敏感内容、用户信息等	删除或清洗敏感数据
日期修正	比如 13 月、2 月 30 号等	清洗为空或就近日期

  - 大模型数据清洗模式，基于语义理解：

异常类型	说明	处理方式
异常截断	非段落换行、文档不完整	清洗换行符，过滤语义不完整数据
拼写错误	同音错别字、键位错别字	修正拼写错误、备份原始数据
特殊内容移除	文本中网页链接、html 等格式标签	移除对应的链接或标签
文档标准化	文档内容不统一，包含繁体字、小语种	文本 Unicode 标准化、繁体转简体，语言翻译

- **常见工具**
  - Data-juicer：提供了开源的数据清洗算子和使用示例，可部署使用。
  - PAI-Designer：支持算子的可视化 pipeline 编排，进行数据加工处理。
- **效果示例**
  - 用户敏感信息过滤：

张三生于 1998 年 12 月 26 日，手机号码为 13200001123

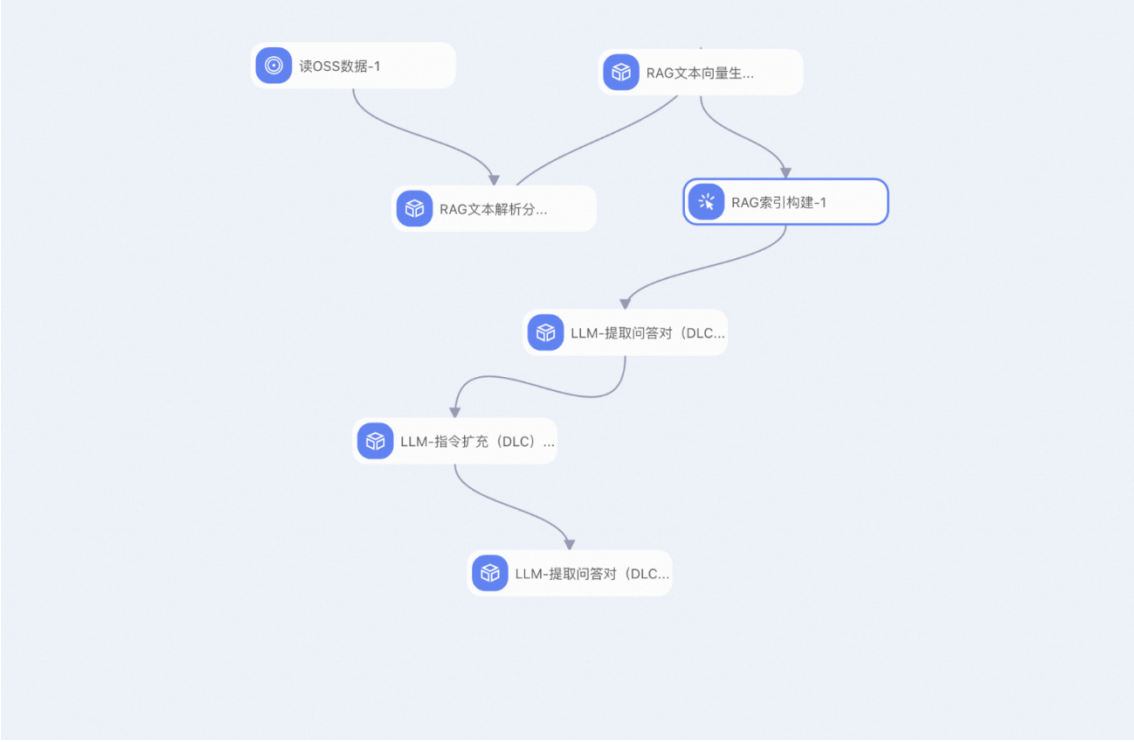
敏感词替换后：

张三生于 1998 年 X 月 X 日，手机号码为 13******23

2. 数据增强

在实际业务中，企业存在可用数据不足、数据分类不均等问题。若清洗后的数据不满足模型训练的需求，数据增强可弥补这类问题。

- **主要方法**
  - 数据裂变：通过大模型合成数据补足。常见方式有：文本重组、文本改写、生成式增强等。
  - 数据均衡：采用数据模拟/同一样本多次采用等弥补低频数据不足问题，增加样本量；高频数据随机欠采样/清洗筛选高质量样本。
  - 鲁棒性处理：通过加入同音错别字、键位错别字、数字口语化，增加训练数据噪声。
- **常见工具**
  - PAI-Designer：选择增强类算子，构建特定场景下的数据增强 pipeline。下图是从知识库中读取数据创建问答对，并进行扩充的示例。



- **效果示例**
  - 数据裂变 - 指令扩充示例：

知识库表述	裂变后表述
订单的发货时效标准	我下单后多久能发货?
	一直没发货怎么办?
	订单一般几天内发出来?
	不同城市、不同快递公司的发货时效有哪些区别?
	有没有当天发/次日达之类的规则?

9.4.2 数据标注

在模型训练中，原始数据通常是非结构化或半结构化的，无法直接用于监督学习或偏好对齐。因此，需要进行数据标注，并筛选标注后的高质量数据进行训练。

- **主要方法**
  - 人工标注：按照任务形式进行分发，由人工对数据进行标注。
  - 自动标注（辅助标注）：由大语言模型/多模态模型驱动，对数据进行辅助标注，再由人工进行审核确认。
- **标注工具**
  - PAI-iTag：支持图像、文本、视频、音频等多种类型的标注，多模态混合标注。同时，提供标注内容组件和题目组件，可直接使用平台预置的模板，也可以根据场景自定义。

9.4.3 数据管理

贯穿整个数据加工链路，包括质量评估、数据归档、数据生命周期管理模块。

1. 应用质量评估

评价后训练模型、智能体应用在实际场景的效果。通过不同评估方式来评测 RAG、模型以及提示词的效果，为迭代优化提供基础。

- **评估方式**
  - 人工评估
    - 专家打分：专家对用户 query 和模型回复，进行人工打分。
    - 用户评价：线上收集用户对模型回复的星级评价，或赞/踩评价。

- 盲测：业务专家对同一问题和不同模型回复，进行对比排序，常用于开放性、主观性场景评测。

- 自动化评估
  - Evaluation 定义：实现对特定场景的回答评估。问答场景代码示例：

```
"evaluation_configuration": {
        "metrics": [ --需要评价的维度
          "question_context_gram",
          "sentimentAnalyzer",
          "answer_context_gram",
          "llmAnswerRelevancy"
        ],
        "evaluation_template": {
          "questionId": "",
          "document": "",
          "question": "", --问题
          "contextTruths": [], --正确上下文
          "documentId": "",
          "answer": "", --模型回答
          "contexts": [], --上下文
          "answerTruths": [] --正确答案
        },
        "enabled": True
      },
....
}
```

评估代码示例：

```
import traceback

from evaluation.core.pipeline.pipeline_rag_qa import Evaluation
from typing import List
from datasets import Dataset
from yic_rag_synthesis.utils.log_util import sls_logger

class EvaluationModel():
    def __init__(self,metrics:List[str]):
        self.evaluation = Evaluation()
        self.metrics = metrics

    def evaluate_data(self,evaluation_template,question,contexts,answer:str):
```

```
        try:
            evaluation_template['question'] = question
            evaluation_template['contexts'] = [item.page_content for item in contexts]
            evaluation_template['answer'] = answer

            data_item = Dataset.from_list([
                evaluation_template
            ])

            sls_logger.info(algo_log=f"evaluation data_item :{data_item},element:{data_item[0]}")

            results = self.evaluation.rag_evaluation(data_item, self.metrics,
sample_evaluation=True)

            sls_logger.info(algo_log=f"evaluation results :{results}")

            score = {}
            if 'hr_doc' in self.metrics:
                score['hr_doc'] = max([item[0] for item in results['retrieval']['hr_doc']['hr_list']])

            if 'mrr_doc' in self.metrics:
                score['mrr_doc'] = results['retrieval']['mrr_doc']['mrr_list'][0]

            if 'answer_rouge' in self.metrics:
                score['answer_rouge'] = round(max([item['gram-l']['rouge'] for item in
                                    results['generate']['answer_gram']['answer_gram_list'][0]]), 2)

            if 'answer_f1' in self.metrics:
                score['answer_f1'] = round(max([item['gram-l']['f1_score'] for item in
                                        results['generate']['answer_gram']['answer_gram_list'][0]]), 2)

            if 'llmAnswerCorrect' in self.metrics:
                score['llmAnswerCorrect'] = results['generate']['llmAnswerCorrect']['llm_list'][0] ==
'CORRECT'

            if 'llmAnswerRelevancy' in self.metrics:
                score['llmAnswerRelevancy'] = results['generate']['llmAnswerRelevancy']['llm_list'][0]
== 'CORRECT'

            if 'answerEmbeddingSimilarity' in self.metrics:
                score['answerEmbeddingSimilarity'] = round(

max(results['generate']['answerEmbeddingSimilarity']['answer_embedding_distance_list'][0]),
2)
```

```
            if 'context_rouge' in self.metrics:
                score['context_rouge'] = round(max([item['gram-l']['rouge'] for item in
                                        results['retrieval']['context_gram']['context_gram_list'][0]]), 2)

            if 'faithFulness' in self.metrics:
                score['faithFulness'] = results['generate']['faithFulness']['faithfulness_list'][0]

            if 'question_context_gram' in self.metrics:
                if len(evaluation_template['contexts']) == 0:
                    score['question_context_gram'] = 1
                else:
                    score['question_context_gram'] = max([max(item['gram-3']['rouge'],item['gram-
l']['rouge']) for item in
results['retrieval']['question_context_gram']['question_context_gram_list'][0]])

            if 'answer_context_gram' in self.metrics:
                if len(evaluation_template['contexts']) == 0:
                    score['question_context_gram'] = 1
                else:
                    score['answer_context_gram'] = max([max(item['gram-3']['rouge'],item['gram-
l']['rouge']) for item in
results['generate']['answer_context_gram']['answer_context_gram_list'][0]])

            if 'sentimentAnalyzer' in self.metrics:
                score['sentiment_neg'] =
results['generate']['sentimentAnalyzer']['sentiment_analyzer_list'][0]['neg']
                score['sentiment_neu'] =
results['generate']['sentimentAnalyzer']['sentiment_analyzer_list'][0]['neu']
                score['sentiment_pos'] =
results['generate']['sentimentAnalyzer']['sentiment_analyzer_list'][0]['pos']

        except Exception as e:
            sls_logger.error(algo_log=f"evaluation error : {traceback.print_exc()}")
            sls_logger.error(algo_log=f"computer scores is error:{e}.")
            for item in self.metrics:
                if item not in score:
                    score[item] = 0
        return score

    def up_langfuse(self,handler,score):
        for name,value in score.items():
            handler.trace.score(
                name=name,
                value=value
            )
```

- 大模型评估：利用 LLM 抽取并比对模型回复与标准答案之间的相关性、准确性。主观开放性问题的建议评价指标如下：

指标	描述说明
准确性	答案描述信息是否准确
可解释性	答案推理或证明是否可解释性
问题召回	问题点是否都回答覆盖
要点信息召回	标准答案中的关键要素是否都覆盖

2. 数据归档

应用上线后会不断产出新数据，这些数据能很大程度反应效果好坏。因此，为了更好地使用这些数据，需要先进行数据归档。

回档后的数据主要应用在 BI 分析、RAG 回流、模型训练迭代。

3. 数据生命周期管理

- **核心目的**
  - 通过数据版本管理，解决过期数据对真实数据的干扰。
- **常见数据时效问题**
  - 数据过期：数据时间超出实际的使用周期，比如新闻过时、产品版本更迭。
  - 数据冲突：两份数据信息相互矛盾，比如信息的勘误与更新。

9.4.4 数据服务

数据服务围绕三类核心数据资产展开，包括数据集、知识库、应用回流。

- **数据集**
  - 数据集的质量、规模及多样性是决定模型性能基线、知识边界和行为范式的关键要素。高质量的训练数据集是保障模型基础能力的基石。
- **知识库**

- 知识库是解决大模型内容“幻觉”现象，并确保信息时效性与准确性的关键机制。通过集成检索增强生成（RAG）技术，智能体能够基于知识库提供精准且可追溯来源的回答。

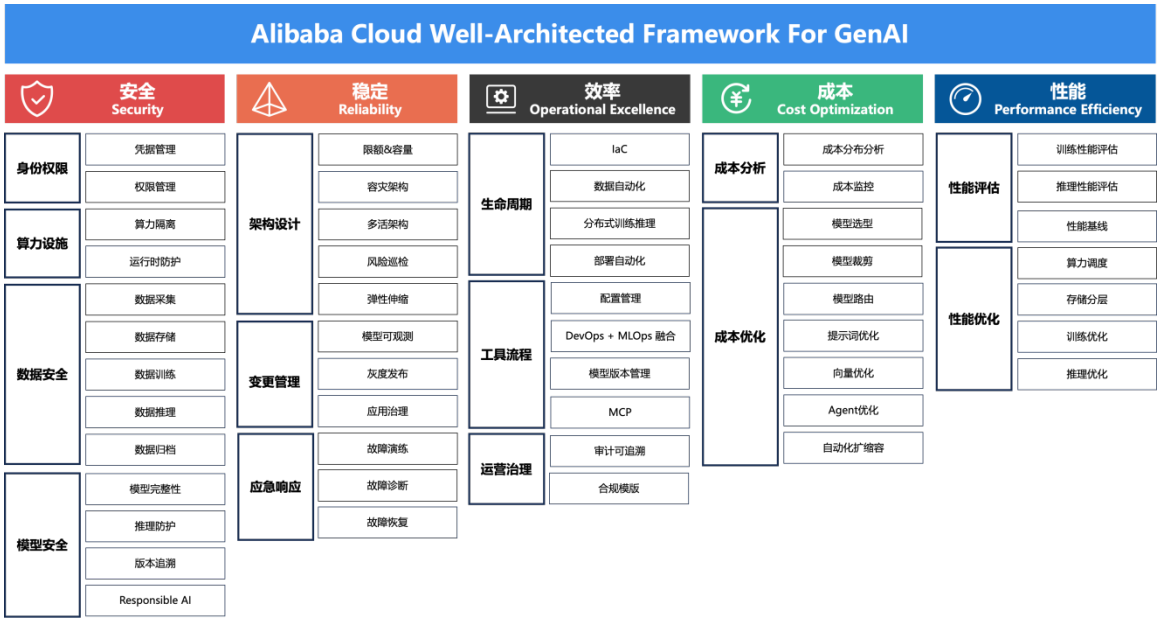
- **应用回流**
  - 这是实现“数据飞轮”并推动智能体自我进化的核心机制。通过对用户交互反馈数据的深度分析，可有效识别模型的潜在不足、挖掘新增业务需求，并将高质量的反馈信息转化为新的标注数据来源，从而实现对智能体效果的持续优化。

第六章．AI 治理

过去几年，阿里云通过卓越架构（Well-Architected Framework）服务了众多不同行业的大型客户，帮助企业在云上构建安全、稳定、高效的架构实践。通过卓越架构框架及评估工具，客户能够发现现有架构与最佳实践的差距，并在专家与合作伙伴的支持下不断迭代优化。这一方法论已成为企业用好云、管好云的重要基石。

随着人工智能尤其是生成式 AI 的迅速崛起，卓越架构框架提出的五大支柱 (安全、稳定、效率、成本、性能)在生成式 AI 时代同样适用。无论是保障 AI 数据的全生命周期安全，确保大模型训练推理的高可用与性能，还是优化 GPU 算力成本与资源效率，这些支柱依旧是企业评估与优化架构的核心维度。

AI Well-Architected Framework



卓越架构框架涵盖的五大支柱（安全、稳定、效率、成本、性能）在生成式 AI 时代，其内涵和实践重点都需要结合 AI 的特性都进行了延展：

- 安全：生成式 AI 涉及的数据来源更为复杂，涵盖个人隐私、企业敏感信息和跨境数据流动。确保数据全生命周期的安全合规，以及模型输出的可信与可解释，成为构建 AI 应用的首要前提。
- 稳定：大模型训练与推理任务往往持续时间长、规模庞大，任何节点的故障都可能导致重大损失。AI 架构需要具备面向失败的设计能力、全链路的容灾方案以及多层次的可观测性，以保证业务连续性。
- 效率：生成式 AI 应用的迭代速度远超传统软件，企业需要新的运维模式，支持多模型协同、快速灰度发布与持续监控，形成从开发到上线的闭环运维能力。

- 成本：AI 场景对 GPU 等高性能算力的需求极为突出，若缺乏有效管理，极易造成资源浪费与成本失控。通过弹性调度、算力池化、Spot 实例与混合精度计算等手段，企业可以在性能与成本间找到平衡。
- 性能：AI 模型规模不断扩展，对存储 I/O、网络带宽和推理延迟的要求更高。通过分布式训练框架、推理加速引擎和边缘侧优化等技术，可以有效提升端到端性能，保障用户体验与业务价值。

客户案例一：某全球运动服饰企业 AI Landing Zone 设计案例

业务背景与挑战

公司背景与 AI 创新探索

某全球运动服饰领军企业，在阿里云上已构建了一套成熟、标准的多账号 Landing Zone 治理体系。随着 AI 时代的到来，该公司敏锐地洞察到 AI 在设计领域的巨大潜力，并与阿里云合作，探索利用“PAI+GPU 算力+通义模型”的全栈解决方案，为服饰鞋履设计环节注入创新活力。

其核心 AI 应用场景是通过 PAI-ARTLAB 训练鞋服设计模型，旨在：

- **提升设计效率**：大幅缩短设计师的创意构思与呈现周期。
- **激发创意灵感**：通过 AIGC 生成多样化的设计方案。
- **优化供应链**：加速设计定稿，缩短订货周期。

这个 AI 设计项目，是该公司集团“AI 转型战略”的先行试点，其成功与否，以及能否在集团统一的云治理框架下安全、合规、高效地运行，至关重要。

在成熟 Landing Zone 体系下的新挑战

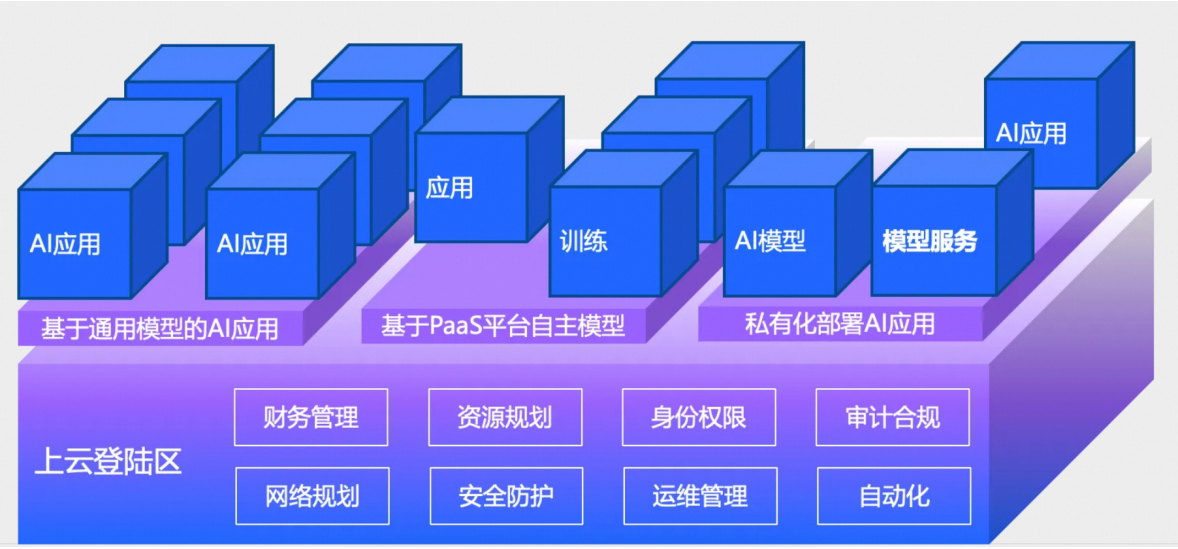
尽管该公司已拥有标准的多账号 Landing Zone，但 AI 业务的引入带来了新的治理挑战，尤其是在**身份权限、财务分账和安全合规**方面，原有的治理框架无法完全适配 AI 平台（如百炼、PAI）的特殊性：

- **身份权限的治理冲突**：按照集团安全策略，所有云上权限都应在中央的**云 SSO** 中统一管理。然而，在当时的技术条件下，PAI 等 AI 平台的工作空间权限管理依赖于在**成员账号**中给 RAM 用户授权，这导致权限管理分散，形成了安全治理的“法外之地”，成为安全部门最大的痛点。
- **财务分账的粒度难题**：AI 应用的成本构成复杂，如何将百炼、PAI 等平台上的模型调用、资源消耗，精准地分摊到具体的业务团队、设计项目甚至单个 API Key，是传统基于账号维度的财务分账模式难以解决的。
- **合规审计的能力缺失**：新兴的生成式 AI 应用带来了新的数据安全与合规风险。原有的合规审计体系，缺少针对 AI 资源（如模型、数据集）的自动化合规检测能力。

因此，核心问题浮出水面：如何在已有的多账号 Landing Zone 体系下，为新兴的 AI 业务‘无缝’地嵌入一套既能满足其独特需求，又符合集团整体治理框架的增强能力？ 这正是构建 AI Landing Zone 的核心目标。

AI Landing Zone 整体架构

本案例中的 AI Landing Zone，其核心并非推倒重建，而是在**通用的、已有的多账号 Landing Zone 基础上，进行能力补齐和强化的企业级云治理框架**。它以阿里云 AI Landing Zone 框架为指导，聚焦于解决客户在“身份权限统一”与“安全合规”方面的核心痛点。



（注：架构图为通用示意图，本案例核心在于治理与集成）

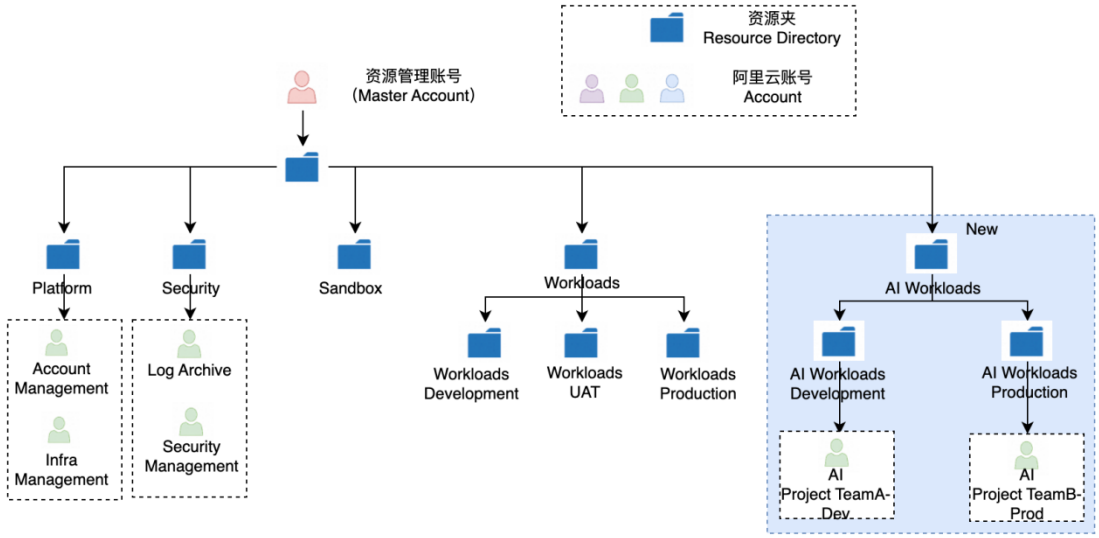
架构以 **百炼、PAI** 为核心 AI 平台，在治理与安全层面，通过 **云 SSO** 、**ActionTrail、配置审计**等服务，实现全链路的操作审计、权限的集中管控和持续的安全合规。

AI Landing Zone 模块详解（增强设计）

本次方案设计的重点，在于针对 AI 业务特性，对现有 Landing Zone 的四大核心模块进行精准升级。

资源管理

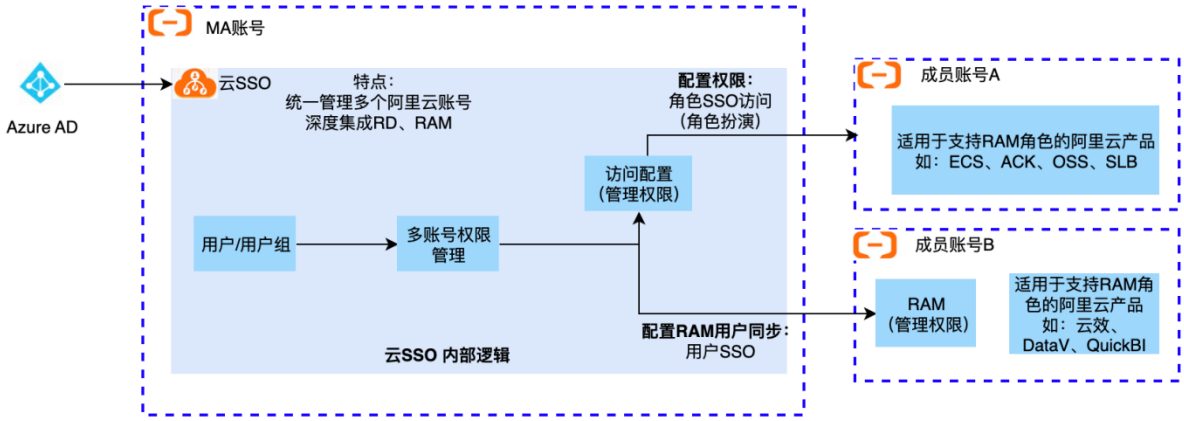
- **现状**：已拥有标准的多账号架构，按功能和业务环境（开发/测试/生产）划分资源夹与成员账号。
- **AI 增强方案：**
  - 在资源目录中，新增一个用于 AI 项目的的 AI Workloads/ AI Workloads Development、 AI Workloads Production 资源夹，专门用于隔离和管理所有 AI 项目。
  - 在资源夹 AI Workloads Development、 AI Workloads Production 下，为 AI 项目组（如设计团队）创建对应独立的成员账号，项目组可以根据自己的需求提出创建测试或者生产阿里云账号。
  - 在成员账号内部，通过百炼或 PAI 的工作空间（Workspace），对更细粒度的应用设计组进行逻辑隔离。
  - 示例架构图如下：



身份权限

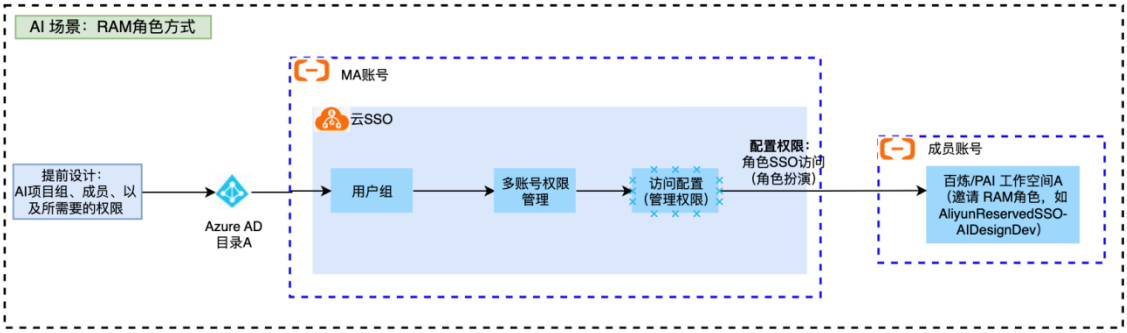
- **现状**:

已通过 Azure AD 与云 SSO 集成，实现了多账号的单点登录。同时使用 RAM 角色和 RAM 用户方式，但受限于 AI/大数据等产品能力，PAI 和百炼产品需要依赖 RAM 用户同步方式去访问，权限管理分散在成员账号，对于该客户是不符合安全规定的。



- **痛点**:
  - 权限分散：RAM 用户权限由成员账号内的业务管理员分配，脱离了云 SSO 的中央管控。
  - 协同困难：PAI Artlab 工作空间中，不同 RAM 用户无法看到彼此创建的数据集，阻碍了设计团队的协同工作。
- **AI 增强方案（核心）**:
  - 逐渐转向 RAM 角色：得益于百炼和 PAI 产品功能的迭代，新方案采用 RAM 角色替代 RAM 用户。

- 权限集中管理：在云 SSO 中为每个 AI 项目组（如 AI-Design-Group）创建对应的访问配置（如，AIDesignDev），其中包含所有必需的云服务权限。
- 统一授权：将 Azure AD 同步过来的用户组，在云 SSO 中直接授予对应 AI 成员账号的访问配置（如，AIDesignDev）。
- 工作空间集成：在 PAI/百炼的工作空间中，直接邀请 RAM 角色（如 AliyunReservedSSO-AIDesignDev）作为成员。这样，项目组所有成员都通过扮演同一个角色访问工作空间，不仅解决了数据集共享的问题，更将权限的最终解释权收归到云 SSO，完美解决了权限分散的痛点。
- 其中，RAM 角色方案的重点在于，事先要设计好 Azure 用户组、用户，以及云 SSO 里该用户组所对应的访问配置（包含名称和所需云服务的权限）。



- **RAM 角色和 RAM 用户两个方案的对比：**
  - 由于不同的客户内部安全要求不同，特对两个方案进行整理比对，用户可以结合内部要求进行选择。
  - 总结：
    - 选择 RAM 角色方案：优先考虑统一权限管理和团队协作效率，适合标准化流程和资源共享场景。
    - 选择 RAM 用户方案：优先考虑操作审计合规性和项目级灵活权限，适合高敏感业务和动态需求。
  - 方案特点、核心优势和差异：

	RAM 角色方案	RAM 用户方案
方案特点与核心优势	**统一管理：**权限通过云 SSO 的访问配置概述集中配置，运维组可全局管控权限策略。 **匿名操作：**工作空间无法显示具体操作人名（需通过什么是操作审计查看具体操作者）。 **团队级共享：**在 PAI Artlab 产品里，不同用户通过 RAM 角色访问时，可共享同一工作空间的资源（如数据集）。	**分散管理：**权限由业务管理者在成员账号的权限策略概览里逐个设置，因此项目组管理者管理权限更加灵活和方式，适合灵活的项目级权限分配。 **实名操作：**工作空间内可直接查看操作人名，满足安全合规要求。 **用户级隔离：**在 PAI Artlab 产品里，不同 RAM 用户创建的资源（如数据集）默认互相不可见。
核心差异点	**统一配置：**权限变更只需在云 SSO 中调整一次，节省 80%运维时间（基于阿里云最佳实践数据）。 **工作空间内匿名：**需要通过操作审计（ActionTrail）查看操作人（需额外配置）。 **低复杂度：**适合标准化流程（如批量任务执行）。 **共享资源特殊场景：**团队成员通过角色访问同一资源（如 PAI Artlab 数据集），提升协作效率。	**分散配置：**需逐个用户分配权限，适合小规模团队（<50 人），但管理成本较高。 **工作空间显示人名：**满足 ISO 27001 等合规要求，无需额外配置。 **高灵活性：**支持细粒度权限（如按用户分配临时凭证），适合动态业务需求。 **用户隔离：**在 PAI Artlab 场景里资源默认私有，需手动共享，适合高敏感业务。

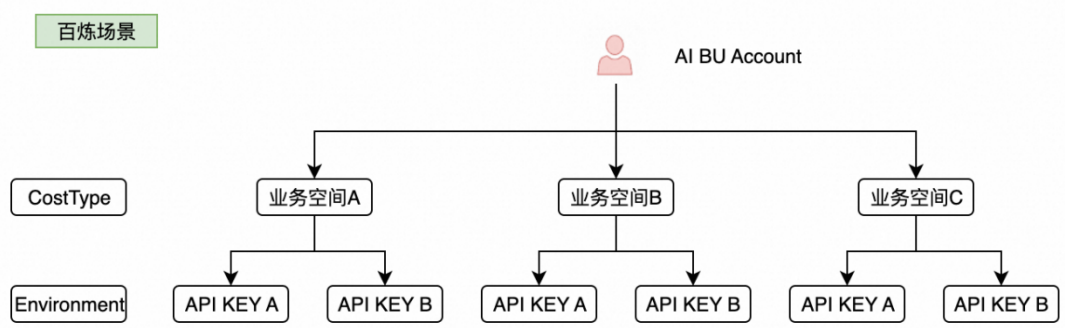
- 典型场景推荐与效率提升

	RAM 角色方案	RAM 用户方案
适用场景	企业级统一权限管理（如跨部门协作）； 截止发稿时，在 PAI Artlab 工作空间中需要数据集共享的场景。	需要工作空间中看到具体用户的敏感业务；或者 AI 项目内需要灵活授权、或精细化权限管理。
具体场景	跨部门统一权限管理等场景	项目级个性化权限；敏感业务操作审计等场景。
解决的问题	● 运维组集中管理权限，避免重复配置，业务组无需介入权限审批； ● 多用户共享同一数据集/模型，避免重复创建。	● 按用户分配不同权限（如只读/编辑），适配复杂项目分工； ● 工作空间显示人名，满足合规要求（如 GDPR、等保 2.0）。
提升效率	● 角色/用户权限配置时间； ● PAI Artlab 的数据集中，资源利用率提升（减少冗余存储和计算）。	● 权限冲突减少（避免角色权限覆盖问题）； ● 审计追溯时间从小时级→分钟级（通过日志自动关联用户身份）。

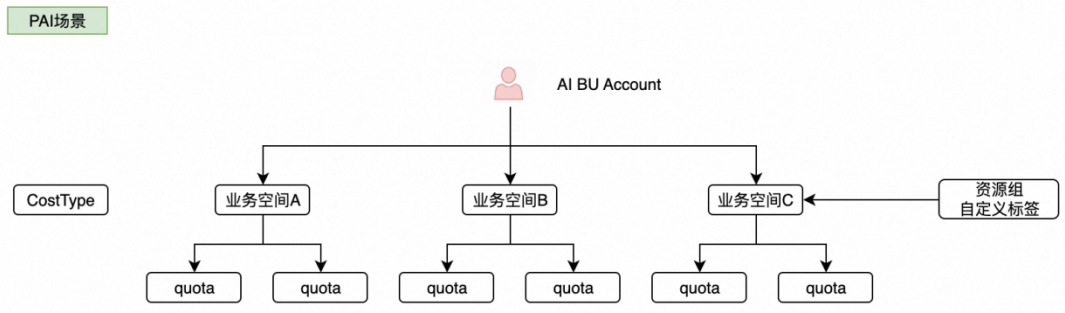
财务分账

- **现状：主要通过财务单元和标签，以成员账号（环境）维度进行成本分摊。**
- **AI 增强方案：**

  百炼场景：建立两级财务单元体系。一级财务单元对应**业务空间**，二级财务单元对应工作空间内的 **API Key**，实现对模型调用成本的精准归因。

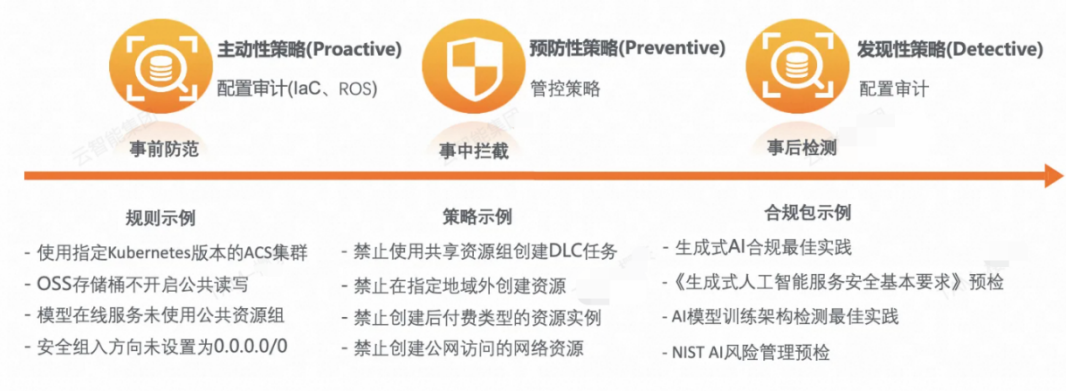


- PAI 场景：采用资源组+标签的方式。为不同团队或项目创建专属的 PAI 计算资源组，并将资源组绑定到指定的工作空间。结合 cost-center 等自定义标签，实现对 PAI 训练和推理成本的精细化分摊。



合规审计

- **现状：已建立事前、事中、事后的通用合规审计体系。**
- **AI 增强方案：**
  - 在配置审计服务中，为所有 AI 成员账号启用阿里云官方提供的“生成式 AI 合规最佳实践”合规包。
  - 该合规包内置了一系列针对 AI 资源的检查规则（如数据隐私、模型训练规范等），能够自动化、持续性地对 AI 环境进行合规检测，主动发现并预警潜在的法律与数据风险。



业务收益

通过对现有 Landing Zone 进行 AI 能力增强，该公司不仅解决了 AI 业务的治理难题，更在多个维度实现了显著价值：

- **资源管理优化，支撑业务高效运行**：独立的 AI 资源夹与成员账号，保障了 AI 业务的资源隔离与弹性，确保了与企业整体云治理策略的无缝衔接。
- **身份权限统一，强化安全合规**：通过云 SSO+RAM 角色的方案，将 AI 项目组的权限统一集中管控，消除了安全风险，完全满足了集团“权限统一管理”的硬性要求。
- **财务分账精细化，驱动成本透明**：通过 API Key、资源组和标签体系，实现了从业务空间到工作空间的分层成本核算，为 AI 投入的 ROI 分析提供了坚实的数据基础。
- **合规审计标准化，降低潜在风险**：通过应用“生成式 AI 合规最佳实践”合规包，实现了对 AI 资源的自动化、持续性合规监控，主动降低了潜在的法律与数据风险。

总结与展望

本案例的成功，为已拥有成熟云治理体系的企业如何引入和管理 AI 业务，提供了一个极具参考价值的范本。其核心经验在于：不颠覆、不另起炉灶，而是在现有 Landing Zone 框架下，通过对关键治理模块（资源管理、身份权限、财务管理、合规审计）进行精准的“外科手术式”增强，实现对 AI 业务的优雅兼容与深度治理。

未来，该公司的 AI Landing Zone 将持续演进，包括探索混合云架构以实现算力灵活调度，将 AI 应用从设计环节深化至供应链、消费者分析等全链路场景，并引入 AI 伦理与零信任架构，进一步提升 AI 系统的可信与安全水平，为数字化转型提供核心引擎。

客户案例二：某新势力汽车品牌智能驾驶训练平台 AI Landing Zone 实践

业务背景与挑战

公司背景与智能驾驶业务

某国内新能源汽车品牌，当前智能驾驶板块重点聚焦于 L2+至 L4 级别系统的研发和交付，已积累大量路测及影子模式数据，用于持续优化智能驾驶模型。

随着业务的飞速发展，AI 模型训练面临着三大核心挑战：

- **数据规模**：达到 PB 级别并持续高速增长。
- **模型复杂度**：Transformer、Diffusion 等大型模型成为主流，算力需求呈指数级增长。
- **算法迭代速度**：为应对复杂交通场景和法规变化，模型需进行快速迭代。

为支撑其技术领先地位，该公司决定将核心 AI 训练业务迁移至阿里云，并以 **AI Landing Zone** 的标准进行规划和建设，其核心需求明确指向：**高性能计算**、**高性能存储**与**企业级治理**。

多维度的企业级挑战

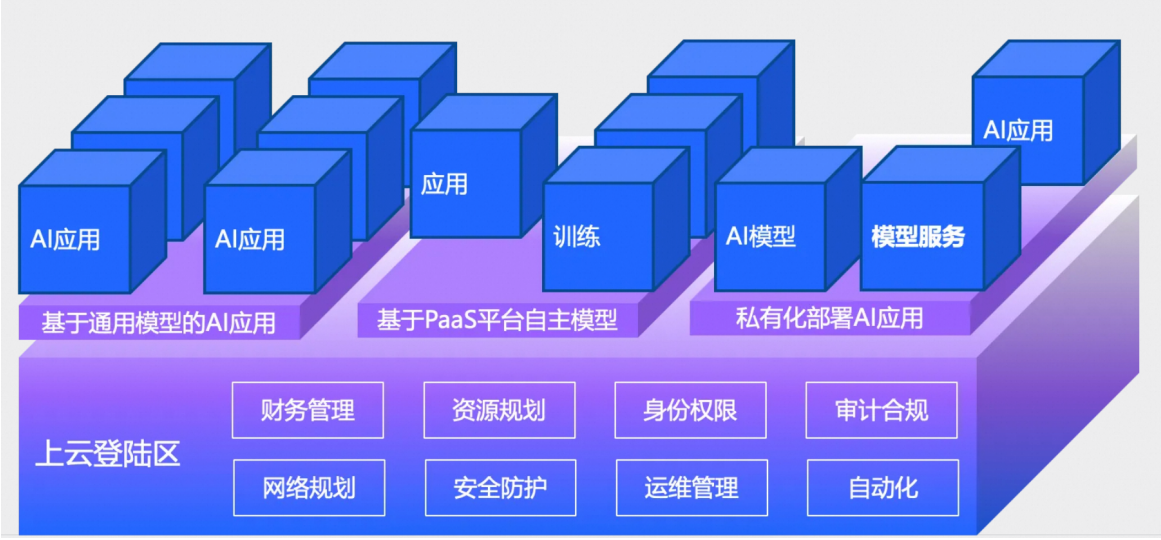
智驾 AI 训练上云是一个典型的复杂项目，它所面临的挑战横跨业务、IT 和安全等多个部门：

- **业务部门（智驾团队）**：核心诉求是**极致性能与效率**。他们渴求稀缺算力资源以加速训练，需要高性能共享存储来消除 I/O 瓶颈，并要求海量路测数据和模型资产得到最高级别的数据安全保障。
- **IT 部门**：核心挑战在于**资源管理与成本控制**。他们需要在统一的云环境下，实现多业务、多环境的资源隔离与配额管理，并对高昂的 AI 算力资源进行精细化的成本归因与分账。
- **安全部门**：核心关注点是**安全合规与权限管控**。必须满足智能驾驶数据安全的严苛监管要求，实现所有云上操作的全链路可追溯审计，并严格遵循最小权限原则，确保核心数据与模型资产的绝对安全。

面对这些盘根错节的需求，一个简单拼凑的云环境远不能满足要求。企业亟需一个体系化的解决方案，既能提供顶级的 AI 工程能力，又能内置完善的治理框架——这正是 **AI Landing Zone** 的核心价值所在。

AI Landing Zone 整体架构

AI Landing Zone 是在通用云采用框架（CAF）基础上，针对 AI 业务特性（如高性能算力、海量数据、模型资产管理等）进行能力增强的企业级云治理框架。本项目基于阿里云 CAF，构建了以 **“安全合规”** 与 **“高性能”** 双轮驱动的 AI Landing Zone 架构。



该架构以 **PAI-DLC/DSW** 为核心 AI 平台，提供弹性、高性能的 AI 训练环境，并使用 GPU 算力作为主要加速资源。**CFPS（高性能并行文件存储系统）** 作为高性能存储，提供 PB 级容量和高吞吐能力。在治理与安全层面，通过 **ActionTrail、配置审计、RAM** 等服务，实现全链路的操作审计、权限管控和安全合规。网络方面，通过 **VPC Peering** 实现与公司现有云环境的网络打通，满足协同开发需求。

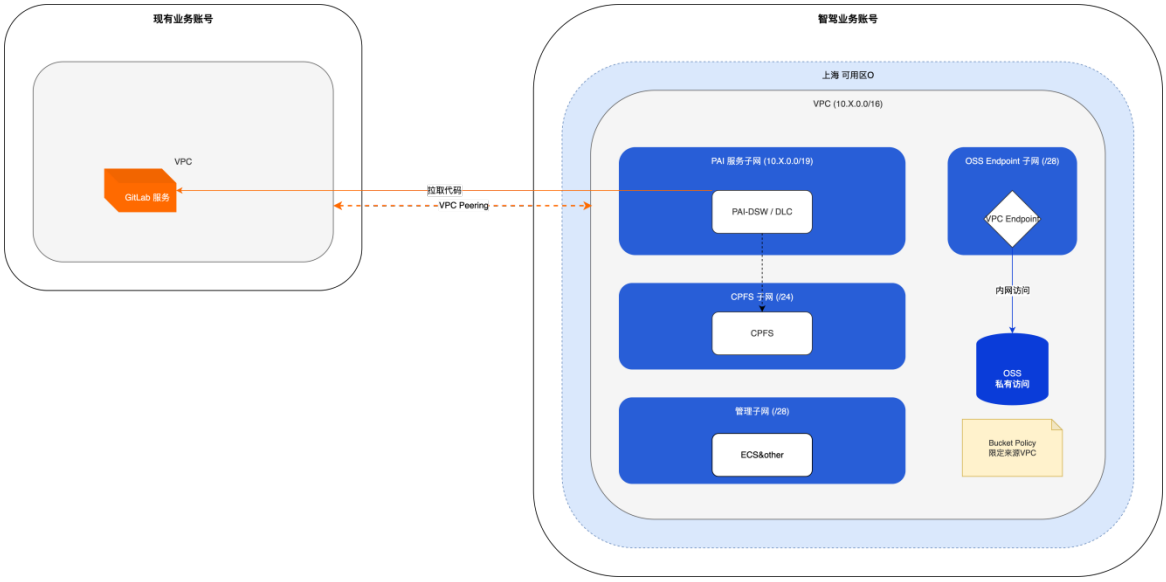
AI Landing Zone 模块详解

资源规划

- **多环境资源隔离**：通过 **PAI 工作空间** 实现业务逻辑隔离，并利用 **资源组** 与统一的 **标签** 策略，实现资源的分类管理、权限控制和成本分摊。
- **PAI 资源配额与伸缩**：针对稀缺的算力，通过 **PAI 资源配额（Quota）** 机制，确保核心业务的算力得到优先保障和预留，并支持弹性使用，最大化资源利用率。

网络规划

- **VPC 设计与网络分区**：采用 **VPC Peering** 方案打通智驾业务 VPC 与现有 VPC 的网络，满足 PAI 训练任务拉取代码等需求。为 PAI 服务规划了独立的 /19 子网段，确保 IP 地址空间充足。
- **安全组与网络 ACL**：严格配置 **安全组** 与 **网络 ACL**，遵循最小化暴露原则。PAI 训练任务通过 **VPC Endpoint** 访问云上存储服务，确保所有数据传输均在阿里云内网进行，杜绝公网暴露风险。



身份权限

- **RAM 角色与权限策略**：通过 **RAM** 实现严格的职责分离。运维团队拥有平台管理权限，智驾团队使用限定操作范围的 RAM 用户，安全团队则被授予只读审计权限。在 PAI 内部，通过工作空间的角色进一步细化权限。
- **跨账号访问控制**：通过 RAM 跨账号授权，实现审计日志的集中管理与安全隔离。

[MISSING IMAGE: 安全审计.drawio.svg, 安全审计.drawio.svg]

安全防护与合规审计

- **安全与合规基线**：启用 **配置审计（Config）** 服务，持续监控并自动修正不符合安全基线的资源配置（如 OSS 禁止公网访问），确保环境持续合规。
- **全链路操作审计**：通过 **ActionTrail（操作审计）** 统一收集所有 API 操作日志，并投递至 OSS 与 SLS 进行长期存储和实时分析。针对 PAI-DSW 内部的审计盲区，通过引入**云堡垒机**作为统一入口，实现对内部高危命令的精细化控制与审计。

[MISSING IMAGE: 云堡垒机纳管 DSW 方案.drawio.svg, 云堡垒机纳管 DSW 方案.drawio.svg]

运维管理与自动化

- **统一监控告警**：基于 **云监控** 和 **SLS** 构建统一监控告警体系，覆盖算力、存储、网络等关键指标，并针对高风险操作配置实时告警。
- **自动化运维（IaC）**：通过 **Terraform** 实现核心基础设施的自动化部署与配置（基础设施即代码），提升交付效率与一致性。

业务收益

通过实施 AI Landing Zone，该公司智驾业务获得了显著的量化收益：

- **效率提升**：高性能算力与存储的结合，使**模型训练任务执行时间减少 25%**，显著加速了算法迭代。
- **成本优化**：精细化的资源配额与成本分摊，提高了资源利用率，使**总体 TCO 降低 10%**。
- **安全增强**：内置的安全与合规体系，成功满足了国家对智能驾驶数据的安全监管要求。
- **运维简化**：基础设施即代码（IaC）的引入，使**运维自动化程度提升 30%**，大幅降低了手动配置的复杂性和风险。

总结与展望

项目成功关键因素

本次项目的成功，得益于三大关键因素：

1. **高层共识与跨部门协同**：项目初期即获得各团队高层共识，明确了职责边界，确保了治理与安全要求的顺利落地。
2. **AI 业务特性优先**：方案设计紧密围绕智驾训练的核心需求，优先保障了算力与 CFPS 存储的极致性能。
3. **遵循云治理基线**：严格遵循阿里云 CAF 框架，以最小权限、全链路审计、配置合规为核心原则，构建了稳固的企业级云平台。

未来演进方向

该公司的 AI Landing Zone 将持续演进：

- **向多账号治理架构升级**：随着业务扩展，将引入资源目录实现更灵活、更安全的组织级治理。
- **引入 AI 观测（AIOps for AI）**：实现对模型训练全链路的智能化分析与优化。
- **全球化网络部署**：随着业务全球化布局，将规划全球化的训练与推理网络方案，支撑海外业务发展。