Qoder

Qoder CLI

终端里的全栈工程师

从 Coding Agent 到企业级 AI 应用基础设施

演讲人：邓忠阳

阿里巴巴 / Qoder技术专家

目录

01

企业级AI应用基础设施

AI应用迫切需要全新的基础设施

当模型足够强、Agent 足够好、环境足够安全，AI 才能真正交付生产力

① 隔离的工作环境

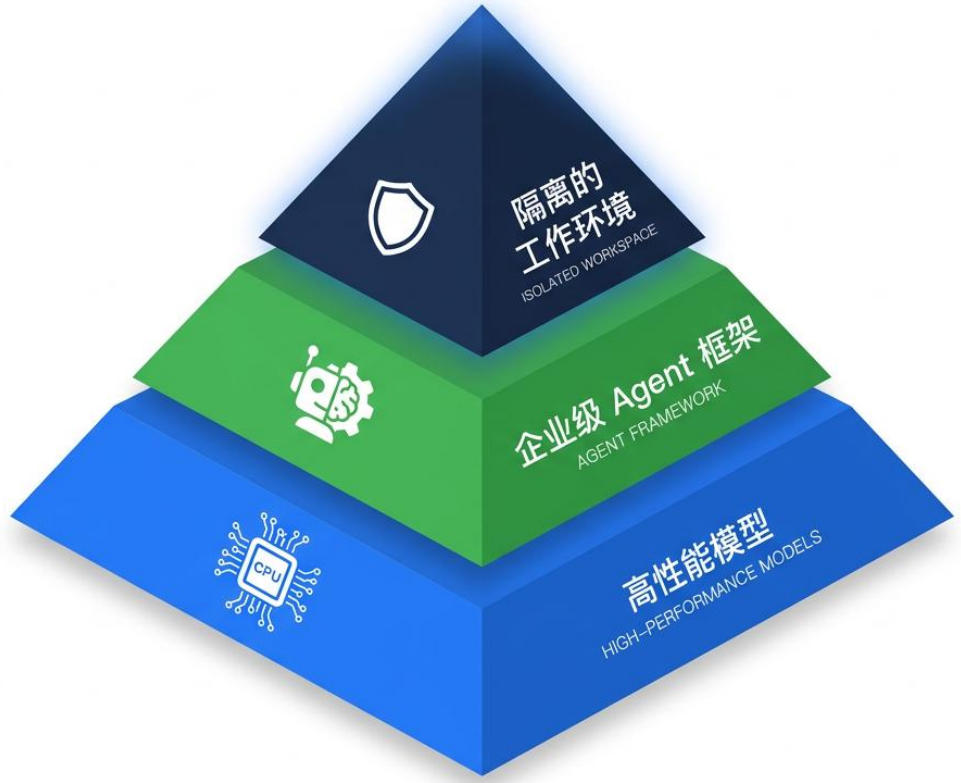
Agent 需要真实的文件系统、终端、网络——但必须与用户环境隔离，支持快照回滚和权限控制，让 AI 放心试错、人类放心放权

② 企业级 Agent 实现

负责把模型的想法变成真实的动作，从“能聊天”到“能做事”的跳跃。包括工具调度、上下文管理、错误恢复、多 Agent 协作——这些工程能力决定 AI 能否可靠落地

③ 高性能模型

智能的上限由模型决定。但是单一模型很难覆盖所有任务，所以需要多模型路由和策略编排



Coding Agent 成为行业赋能的关键

通用的执行底座 —— 把自然语言需求转换成可执行动作

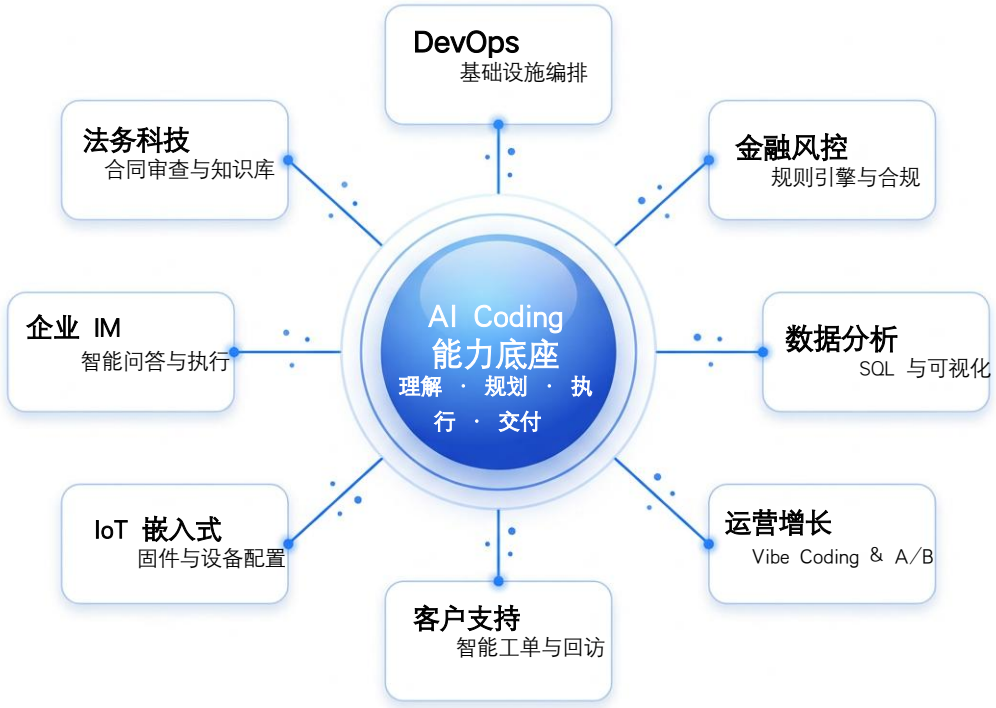
为什么 AI Coding 能赋能各行各业？

AI Coding 的本质是：理解意图 → 规划步骤 → 调用工具 → 交付结果。

这套能力可复用到任何「理解需求+自动化执行」的数字化场景。

各行各业的落地场景：

- DevOps　基础设施编排 · 故障自愈 · 监控自动化
- 金融风控　规则引擎配置 · 报表生成 · 合规检查
- 数据分析　SQL 生成 · ETL 搭建 · 可视化报表
- 运营增长　活动页面 Vibe Coding · A/B 实验自动化
- 企业 IM　智能问答 · 任务执行 · 知识库管理
- IoT 嵌入式　固件脚本生成 · 设备配置批量下发



三层基础设施 × 三款 Qoder 产品

模型能力 + Agent 框架 + 运行环境，一站式企业级AI基础设施

需要多模型支撑

Qoder Models

Model Inference

- 内置海内外前沿模型，开箱即用
- 智能路由 + 策略分级，性能与成本平衡
- 支持 BYOK，企业密钥灵活接入

需要生产级 Agent

Qoder CLI

Agent Framework

- 生产级 Agent 引擎，稳定可控
- 细粒度工具权限 + 审计回放
- 一行 SDK 接入，嵌入任意应用

需要独立运行环境

Cloud Agents

Secure Environment

- 安全沙箱，隔离执行边界清晰
- 快照式会话隔离，状态可恢复
- 弹性算力，按任务自动伸缩

一套 CLI · 三层能力 · 让每个产品都能内嵌 Agent

Qoder CLI 和 Qoder 产品家族

一套 CLI 内核 · 支撑 Qoder 全产品家族

For Developers · 开发者侧

For Knowledge Workers · 工作场景

Qoder

IDE

JetBrains Plugin

IDE 插件

Cloud Agents

云端 Agent

QoderWork

办公场景

QoderWake

数字员工

Qoder CLI

Coding Agent · 工具集 · SDK · 沙箱接口

大语言模型 LLM Layer

编排 统一调度多 Agent，跨产品复用能力

集成 SDK 一行接入，快速嵌入 Agent 能力

进化 共享内核，一处升级，全家族受益

02

Qoder CLI 技术架构解析

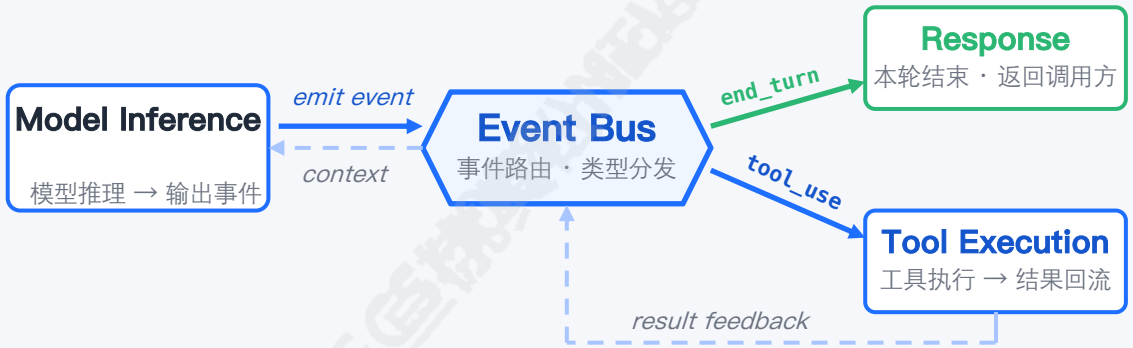
Qoder CLI 整体技术架构

三层主干 + 两根横切：一个 Agent Loop 为核心，被知识层供给、被可观测层监控

基于事件驱动的 Agent 执行循环

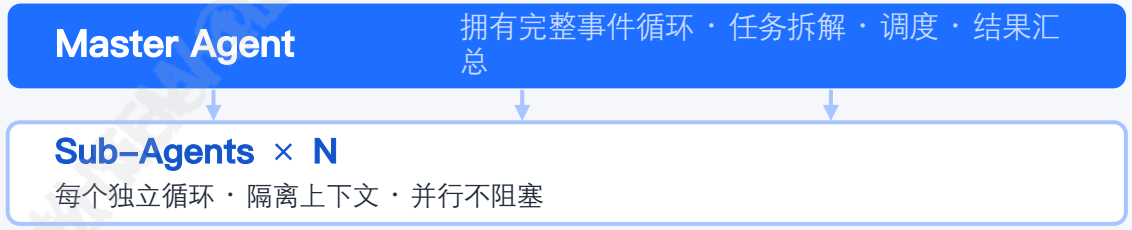
模型输出即事件 · 事件类型决定路径 · 同一个循环 × N 个 Agent

A. 单 Agent 内部 · 事件循环



↓ 同一个循环 × N 个 Agent ↓

B. 在循环之上 · 主 / 子 Agent 派生




```python
# event_loop.py
Master Agent Loop · 伪代码
while True:
    event = model.infer(context)

    if event.type == "end_turn":
        return event.response   # 本轮结束

    elif event.type == "tool_use":
        result = execute(event)
        context.append(result)  # 结果回流
        continue
```

为什么用事件驱动？

✗ **同步函数调用**

模型 → 工具 → 模型，调用链耦合死

✓ **事件驱动**

模型只产事件，调度由 Event Bus 决定

→ 可插拔 · 可观测 · 可并发

解耦的力量 模型、工具、宿主三者解耦 — 加工具、加审批、加审计、加子 Agent 都有稳定扩展点

面向复杂长程任务的上下文管理机制

隐式系统提醒避免信息遗忘 + 智能压缩保证长程任务持续执行不中断

1 Reminder · 隐式系统提醒

▸ 触发时机：每次推理前

每一轮都注入：当前任务状态 · 工具结果 · 文件变更 · 关键路径

▸ Reminder 类型 · 4 类分组

1 执行状态 工具状态 · 文件变更	2 规划 Plan · Todo
3 能力 Skill · Sub-Agent	4 记忆 Memory

→ 数百轮后仍能**准确感知**任务目标 · 中间决策 · 关键路径

2 Compression · 系统压缩

▸ 触发时机：上下文使用率接近阈值

自动触发 或 用户手动 /compact

▸ 多层递进方案

Session Memory
完整对话历史
→ **Auto Summary**
智能摘要压缩
→ **Micro Memory**
关键信息提炼

→ **保留**关键决策 · 工具结果 · 代码片段 · 错误信息 · **砍掉冗余**

长程任务不是靠模型硬扛上下文，而是靠 Reminder 主动注入 + Compression 智能压缩 共同维护任务连续性

03

Qoder CLI SDK能力解析

数行 SDK 代码 完成 Agent 能力集成

提供 TypeScript / Python SDK · 把完整 Agent 能力开放给所有开发者

example.ts · 最小可运行示例 TypeScript

```typescript
import { accessTokenFromEnv, query } from '@qoder-ai/qoder-agent-sdk';

for await (const message of query({
  prompt: 'Analyze the codebase, find functions without
   test coverage, and write unit tests for them.',
  options: {
    auth: accessTokenFromEnv(),
    allowedTools: ['Read', 'Write', 'Edit', 'Glob', 'Bash'],
    permissionMode: 'acceptEdits',  // 自动确认文件修改
  },
})) {
  if (message.type === 'assistant') {
    for (const block of message.message.content) {
      if (block.type === 'text')  console.log(block.text);  // AI 回复
      else  console.log(`Tool: ${block.name}`); // 工具调用
    }
  }
  else if (message.type === 'result') {
    console.log(`Done: ${message.subtype}`); // 完成
  } }
```

集成只要 4 步

与左侧代码 4 个区域一一对应

1. **导入 SDK**
   `@qoder-ai/qoder-agent-sdk`
2. **给 prompt + options**
   `tools · permission · auth`
3. **for await 流式消费**
   `text · tool_use`
4. **拿到 result**
   subtype = success → 任务完成

Agent 能力不是黑盒 CLI，而是 SDK 是 积木 — 任何产品都能嵌入完整 Agent 能力

模型使用策略 集成方自主决策

固定模型 · 动态选择，业务侧按场景灵活控制每次 LLM 调用

▸ 4 档模型矩阵

ultimate	performance	efficient	lite
最强	平衡	高效	极简
复杂推理 · 多步规划	日常开发 · 主线编码	大批量调用 · 流水线	简单查询 · 低延迟
能力 ●●●●	能力 ●●●○	能力 ●●○○	能力 ●○○○
速度 ●○○○	速度 ●●●○	速度 ●●●●	速度 ●●●●
成本 ●●●●	成本 ●●●○	成本 ●●○○	成本 ●○○○

1 固定模型

▸ **适用** · 场景稳定 · 调用成本/质量已知 · 无需运行时调整

fixed.ts

```ts
import { qodercliAuth, query } from '@qoder-ai/sdk'

const q = query({
  prompt: 'Analyze this code',
  options: { auth: qodercliAuth(), model: 'performance' }
})
```

代码最简 · 一行选档 → 上线后行为高度可预期

2 动态选择

▸ **适用** · 不同任务用不同档位 · 成本与质量动态权衡

dynamic.ts

```ts
import { type ModelPolicyProvider } from '@qoder-ai/sdk'

const policy: ModelPolicyProvider = ({ purpose }) => {
  if (purpose === 'planning') return { model: 'ultimate' }
  return { model: 'performance' }
};
```

规划用 ultimate · 编码用 performance → 按需调配，成本可控

SDK 不替你选模型 — 4 档摆出来，集成方按场景自主决策 · 不锁定，成本可控，质量可调

模型使用策略 SOTA 模型随意切换

三种模型来源协同 · 集成方不被单一供应商锁死

① Auto 智能档

SDK 内置 5 档 · 智能调度 · 无需关心模型名

策略池

- **Auto** · 推荐 — 智能选档 · 均衡成本 — 1.0x
- **Ultimate** · 最强 — 专家级深度推理 · 峰值质量 — 1.6x
- **Performance** · 平衡 — 高级推理 · 高质量输出 — 1.1x
- **Efficient** · 高效 — 标准推理 · 低成本执行 — 0.3x
- **Lite** · 极简 — 基础推理 · 免费可用 — Free

② 前沿池 · 持续追新

厂商发新模型 · 自动接入 · 无需自管订阅

SOTA 系列

- **Qwen-Plus/Max** — 通义千问 · 增强推理与 Agent 可靠性 — AUTO UPDATE
- **DeepSeek-V4** — DeepSeek · 卓越 Agent 能力与推理 — AUTO UPDATE
- **GLM 系列** — 智谱 · 长程任务工程级输出 — AUTO UPDATE
- **Kimi K 系列** — 月之暗面 · SOTA 编码与 Agent 群控 — AUTO UPDATE
- **MiniMax M 系列** — MiniMax · 自进化模型迭代能力 — AUTO UPDATE

③ BYOK · 自带密钥

对接三方服务 · 走自己账户 · 合规可控

三方服务商

- **阿里云百炼** — *Bailian* — 通义千问全系列模型 — API KEY
- **智谱 Z.ai** — *Z.ai* — GLM 全系列模型 — API KEY
- **月之暗面** — *Moonshot* — Kimi 系列模型 — API KEY
- **MiniMax** — *MiniMax* — MiniMax 自研系列 — API KEY
- **DeepSeek** — *DeepSeek* — DeepSeek 全系列模型 — API KEY

一套 SDK 接入三种模型来源 — 自主决策 · 不锁死 · 应对丰富多样的商业模式

五道关卡实现细粒度权限控制

从静态规则到运行时审批，全方位管控 Agent 工具能力

Tool Call → ① 工具范围 → ② 权限模式 → ③ 运行时审批 → ④ 声明式规则 → ⑤ Hooks 审计 → ✓ Execute

① 工具范围

`allowedTools`
`disallowedTools`
静态白/黑名单 · 一刀切

② 权限模式

`plan / auto`
`dontAsk / bypass`
一次性预设模式 · 全局生效

③ 运行时审批

`canUseTool(tool, input)`

回调宿主 · 可注入审批 UI

④ 声明式规则

`settings.permissions`

项目级配置 · 团队共享

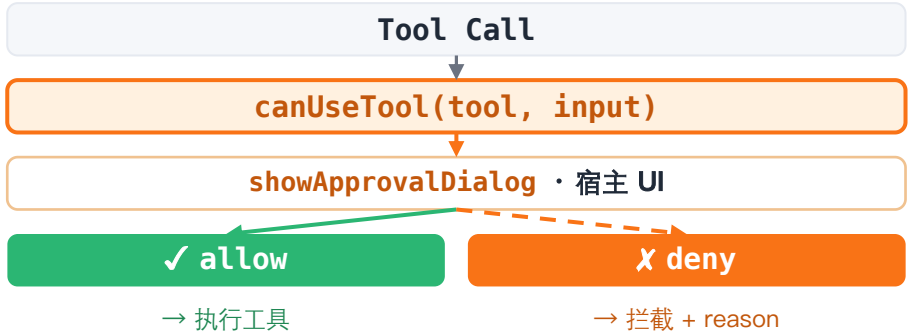
⑤ Hooks 审计

`PreToolUse`
`audit`
细粒度日志 · 行为可追溯

③ canUseTool · 把决定权交给宿主应用

approval.ts

```ts
query({ prompt, options: {
  permissionMode: 'default',
  canUseTool: async (tool, input) => {
    // 把审批请求展示在宿主 UI 中
    const ok = await showApprovalDialog({ tool, input });
    return ok
      ? { behavior: 'allow', updatedInput: input }
      : { behavior: 'deny', message: 'Rejected' };
  }
}})
```



五层纵深防御 — 细粒度可观可控 · 满足企业风控合规要求

灵活扩展、按需组合的插件体系

五类扩展点相互正交，可独立使用也可协同工作，构建专属 AI 工程师

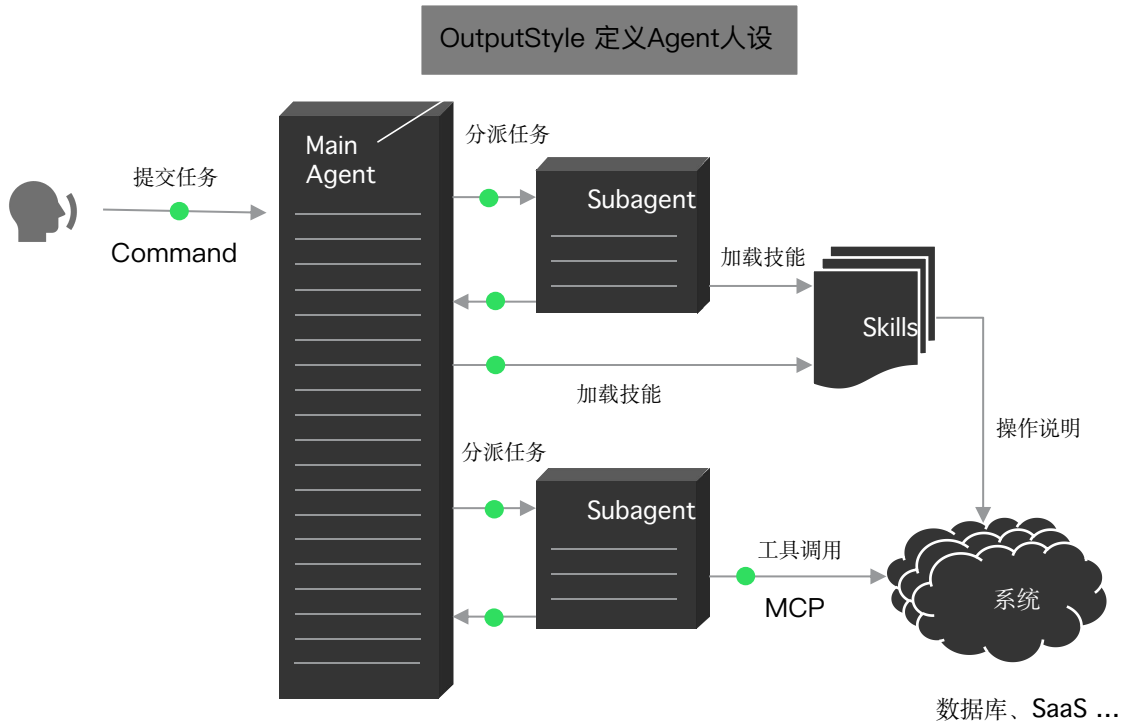
① **Output Style** — 输出层
自定义 AI 输出风格 · 适配团队表达规范
`system prompt · Markdown / JSON / 表格 / 自然语言`

② **MCP** — 输入层
接入任意外部工具与数据源 · 无限扩展 Agent 能力边界
`数据库 · API · 监控系统 · 公司内部系统`

③ **Subagent** — 编排层
按场景创建专属子 Agent · 各司其职并行协作
`安全审计 Agent · 测试 Agent · 评审 Agent`

④ **Skill** — 流程层
封装可复用工作流 · 一行 /slash 命令触发复杂任务链
`/release · /review · /onboard · 团队内共享最佳实践`

⑤ **Hook** *· 通用拦截器* — 拦截层
在工具执行前后注入自定义逻辑 · 通用拦截器
`PreToolUse · PostToolUse · 安全扫描 · 审批 · 日志`



● 示例Hook点位，支持数十种类型Hook操作

04

典型集成场景

集成场景

第一幕 · 救火

跨境支付应急 · 22 分钟闭环

凌晨 02:17 · 跨境清算链路告警 · 自动值守 Agent 接管

「凶手不是 DB，是清算域少了一行配置。」

— 误判 DB 锁 → 跨域复核 → 根因平反 → 写回应急平台

▸ 时间轴 · 22 分钟 4 个关键节点

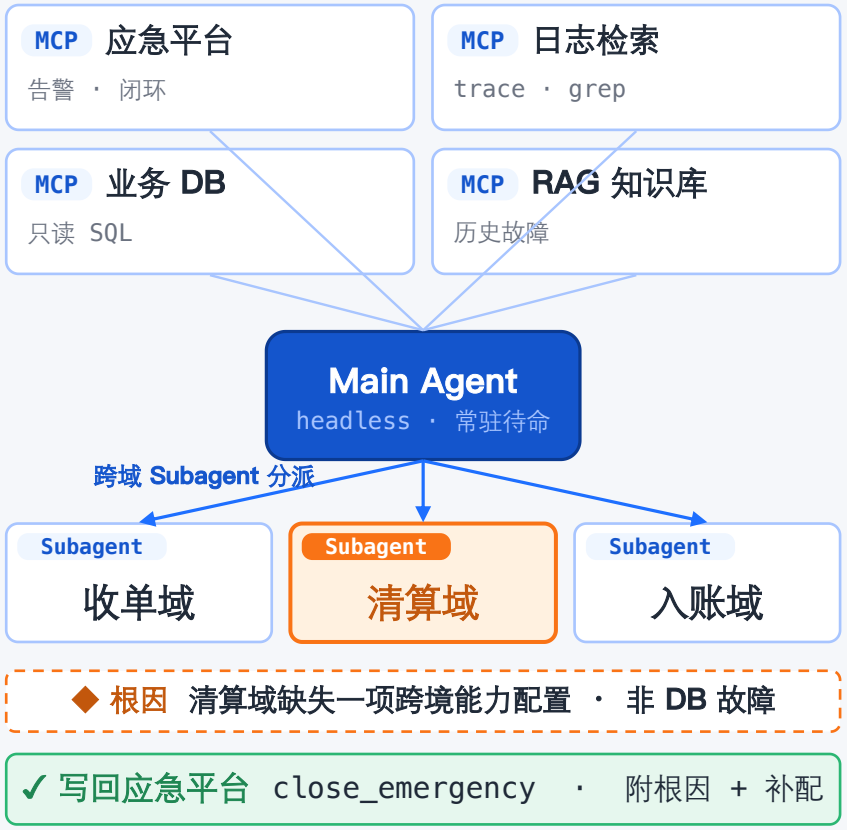
告警涌入	拉取证据	跨域复核	闭环写回
T+00	30 trace · 18 机器 T+08	三 Subagent 串行 T+15	根因 + 补配动作 T+22

22 min
闭环耗时
vs 人工 ~2h

0 笔
资损
无误付 · 无延迟

30 trace
证据规模
× 18 机器 · MCP 聚合

▸ 集成姿势 · 这一幕用到了什么



◆ 根因 清算域缺失一项跨境能力配置 · 非 DB 故障

✓ 写回应急平台 close_emergency · 附根因 + 补配

集成姿势 — MCP 接企业系统 · 跨域 Subagent 分派 · headless 容器常驻待命

集成场景

第二幕 · 值守

顶会理论实验 · 10 小时无人值守

周五下班离开工位 · durable Cron 接管 · 次日早晨实验全 PASS

「会话退出了，实验还在跑。」

— 写 POC → 提交训练云 → Cron 轮询 + 失败重试 → 3/3 PASS

▸ 时间轴 · 10h10m 跨夜值守 4 个关键节点

POC 落盘	提交训练云	OOM 自愈	全 PASS
T+00:00	T+00:30	T+05:00	T+10:10

10h 10m
无人值守
0 人盯 · 跨夜

3 / 3
理论 PASS
首轮即达成

1 次自愈
OOM 重启
降 batch · 续跑

▸ 集成姿势 · 这一幕用到了什么

durable Cron 定时唤醒 Agent
会话退出也能跑 · 断点续推 · 跨夜不掉

MCP 训练云
提交 · 轮询

state 进度文件
qoder_progress.json

state 失败 ledger
OOM · 重启日志

MCP RAG 知识库
理论 / 配方

Main Agent
headless · 持久状态机

◆ 自愈 OOM → 降 batch → 重启续跑

✓ 3/3 理论 PASS lr=6e-7 · 进度文件归档

集成姿势 — durable Cron 唤醒 · 进度文件持久化 · 训练云 MCP 调度

集成场景

第三幕 · 远征

端到端仿真评测 · 多 GPU 远征

SDK 嵌入业务调度器 · 多 GPU 切片 · 仿真场景 rollout 自动评测

「不是工具，
是同事。」

— SDK 嵌业务 → Cron 切片 → 多 GPU 远征 → 视频回传归档

▸ 时间轴 · 一次评测远征 4 个关键节点

SDK 接入	场景注册	GPU 远征	视频归档
嵌业务调度器	10+场景 · 切片	多GPU 并发 rollout	回传 + 自动评分
Day 0	Day 1	Day 2	Day 3

N GPU
并发远征
切片 · 隔离

12 场景
一次评测
rollout · 全程录像

1 条 SDK
嵌入调度器
Qoder 成为同事

▸ 集成姿势 · 这一幕用到了什么

SDK 业务调度器 · Agent 嵌入为同事
Recurring Cron 触发 · 按 GPU/进程切片场景

SDK 嵌入业务进程

Qoder Agent
被业务进程调用

Cron 切片 · 多 GPU 远征

GPU-0..2	GPU-3..5	GPU-6..7
Shard A	Shard B	Shard C
城市道路	高速路况	极端场景

◆ 产出 rollout 视频回传 · 自动评分 · 归档

✓ 调度器收单 eval_report · 继续下一轮远征

集成姿势 — SDK 嵌入业务调度器 · Recurring Cron 切片 · 多 GPU 并发远征

Qoder CLI 能力总览 极致轻量的 AI 编程智能体 · 底座 · 发动机 · Agent OS

macOS | Linux | Windows

核心架构

业界领先
多智能体架构

- 多模型架构：通过主子 Agent
- TUI / Headless 双命令行交互方式
- 多模型动态路由 · 200K 上下文窗口
- 多智能体架构：子智能体调度编排
- 灵活扩展：MCP（工具扩展）+ Skills
- 三方集成：SDK、ACP、Headless 三种集成方式
- 企业级安全、管控，细粒度权限控制 · 沙箱隔离

方法论体系

红线 通过 AGENTS.md 约束执行边界

Skills 注入私域知识 · 消除幻觉

Hooks Agent循环确定性检查点

SPEC 驱动需求→设计→代码→测试

异步委派 任务委派远端，充分释放AI潜力

工具是手段 · 方法论是灵魂

产品家族

CLI Qoder CLI
底座 / 发动机 · Qoder Agent 核心引擎

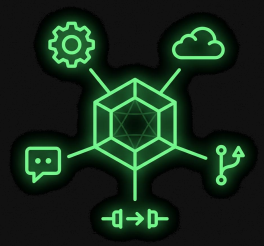
Qoder IDE
面向专业开发领域 · AI 原生 IDE 产品

QoderWork
面向泛工作场景 · 非开发者 AI 工作助手

IDE / QoderWork 是 CLI 不同场景下的 GUI 形态

被集成形态

SDK	TS / Python 语言， 数行代码轻松集成
Headless	执行过程无界面，适配 CI/CD 流水线
ACP	标准 ACP 协议实现，对接各类客户端



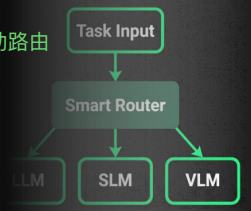
- 自定义 Subagent 扩展业务
- 标准 MCP 协议接入外部服务
- 任意工具细粒度权限控制
- 轻松定义 Agent 执行行为
- 支持打包成任意产品形态

专业 AI Coding Agent 实现，轻松集成先进能力

多模型路由

LLM	SLM	VLM
复杂推理	简单抽取	图片理解

- ✦ 一站式整合海内外 SOTA 大语言模型
- ✦ 支持个人 / 企业级 BYOK，按需接入
- ✦ 按任务类型 / 复杂度 / 成本自动路由
- ✦ 模型不可用时支持自动降级
- ✦ 统一信用计费，不绑定单一模型
- ✦ 不绑定单一供应商，支持多云部署



AI 友好基础设施

从人类工具思维到 AI-Native 思维

- 一条命令快速安装，无任何开发平台依赖
- 适配主流操作系统、Docker、Kubernetes
- 云上 Agent 7×24 独立工作，异步执行
- 本地、云端随意切换，满足不同场景任务诉求
- 分布式架构执行，并行提升 Team 任务效率
- 自由、灵活高度扩展，轻松接入传统基础设施

注：支持 macOS、Linux、Windows 及容器、K8s 环境

应用场景

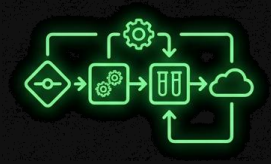
代码智能审查 CI/CD · PR 评论	DevOps 自动化 部署 · 监控 · 日志
Spec 驱动开发 Quest Mode 全流程	Vibe Coding 平台 钉钉开放平台
故障自动诊断 运维流水线	IM 智能机器人 对客答疑 · 助手
远端异步协作 Git → 审查 → MR	冲突检测修复 沙箱隔离 · 闭环

竞品优势

vs Claude Code
多模型 · 不绑定单一供应商 · 企业安全

vs Cursor / Copilot
不绑 IDE · 可被集成 · Headless CI/CD

vs 开源方案
企业支持 · 统一计费 · 完整方法论



打造企业级 Agent 实现的核心引擎

THANKS

Qoder