构建面向 .NET 的 NativeAOT 友好的智能体运行时

OpenClaw.NET：自托管、可观测、默认安全的 .NET AI Agent 运行时与网关

定位： 让 .NET 生态的每位开发者，都拥有安全、可观测、自托管的 AI Agent 运行时。

使命关键词：以 NativeAOT 实现极致启动速度与内存效率，让 Agent 像 Web 应用一样被托管、被运维、被治理。

MIT 开源 · github.com/clawdotnet/openclaw.net

自我介绍

张善友

微软MVP

广州智用开物人工智能科技有限公司 CTO

运营微信公众号“dotnet跨平台” 和 “新一代智能应用“。有25年的大型企业应用开发与架构经验，曾在腾讯工作12年，目前在广州智用开物人工智能科技有限公司担任 CTO，主导企业级“领航navi” 智能体团队平台的开发，精通.NET技术生态，深入掌握CLR、多线程、设计模式，主导多个开源项目。深入大模型应用的开发，最近正致力于开发和推广 新一代应用 的开源项目OpenClaw.NET。

运营微信公众号 ：dotnet跨平台 和 新一代智能应用

@geffzhang

https://github.com/geffzhang

https://www.cnblogs.com/shanyou

02 .NET 生态不缺模型与工具，缺的是一个原生、可托管的 Agent 运行时基座

现 象

Agent 正在从演示走向生产

需要 7×24 在线的运行时、调度与网关，而非一次性脚本

团队已有大量 .NET 资产

引入第二套运行时意味着第二套部署、监控与技能栈

数据合规要求持续收紧

Agent 必须与数据同域部署，自托管从偏好变为硬约束

六 类 痛 点

01 **碎片化**
工具、频道、模型各自为政，无统一运行时

02 **黑箱化**
决策过程不可见，无法审计与复盘

03 **安全恐惧**
工具权限粗放，不敢放开自动化

04 **部署困难**
依赖繁重，难以嵌入现有 .NET 设施

05 **生态孤岛**
.NET 开发者缺少原生 Agent 生态位

06 **可观测性弱**
指标、日志、审计分散，故障定位困难

痛点指向同一空缺：.NET 需要一个 NativeAOT 友好、可自托管的 Agent 运行时。

03

产品定位：AI 时代的 IIS / NGINX —— 托管 Agent，如同当年托管网站

IIS

NGINX

托管了 Web 应用的时代

OpenClaw.NET

托管 Agent 的时代

自托管

Agent 与数据同域部署，基础设施完全自主可控

可观测

指标、日志、审计内建于运行时，决策全程可见

默认安全

最小权限起步，随环境成熟度渐进硬化

.NET 原生

C# 与 dotnet CLI 全链路，不引入第二技术栈

一句话主张

让 .NET 生态的每位开发者，都拥有安全、可观测、自托管的 AI Agent 运行时。

愿景写进架构：NativeAOT 极致启动、五层客户端、渐进安全与被动治理。

OpenClaw.NET 要成为 .NET 生态运行 Agent 的基础设施默认选项。

04

六类核心用户的公约数：在自己掌控的 .NET 基础设施上运行可信赖的 Agent

01 **.NET 全栈开发者**

在熟悉的技术栈中构建、调试与迭代 Agent，无需切换语言生态

02 **运维与 SRE**

需要指标、日志、审计三件套，以及可控、可回滚的自动化

03 **智能家居爱好者**

自托管网关连接家庭设备与消息频道，数据不出家门

04 **企业 IT 管理者**

合规自托管、组织级策略、角色权限与完整审计链

05 **AI 创业者**

以最低边际成本快速搭建并交付 Agent 产品

06 **.NET 生态贡献者**

参与技能、插件与频道适配的生态共建

从个人开发者到企业 IT，诉求收敛为同一件事：可控、可信、可运维的 Agent 基座。

05

六大核心价值：从 .NET 原生体验到对既有 Agent 生态的兼容复用

01 **.NET 原生体验**

C#、dotnet CLI、NuGet 全链路，开发调试零割裂

02 **默认安全、渐进硬化**

Compatibility → Supervised → Lockdown 三级安全姿态

03 **完整运维面**

五层界面覆盖开发、运维、管理与终端用户

04 **80+ 原生工具**

覆盖文件、Web、数据库、智能家居、Canvas 等 15 个工具域

05 **12 种消息频道**

国际 IM 与飞书、钉钉、企业微信统一接入

06 **生态兼容**

复用 SKILL.md 技能与 TypeScript 插件等既有生态资产

价值不是单点功能，而是一条从开发体验到生产运维的完整链路。

06

护城河：NativeAOT、被动治理、零成本压缩

01

业界唯一 NativeAOT Agent 运行时

CLI 与 Gateway 均为 NativeAOT 制品，亚秒级启动、低内存常驻，Agent 可以像系统服务一样被托管

亚秒级

冷启动时间目标

02

唯一 Passive Governance 治理

Harness 四层治理链：意图、事实、决策、执行全程留痕；永不自动生效，审查优先于效率

4 层

Contract → PEV 治理链

03

TokenJuice 零成本上下文压缩

规则驱动、确定性压缩，不调用 LLM；在多轮长会话中压缩效应逐轮放大

50 – 95%

上下文压缩率区间

性能、治理、成本三条线同时建立壁垒，且都来自架构层而非功能层。

07

NativeAOT 是一等架构约束：Core 层零动态反射，JIT-only 功能显式隔离

设计约束

Core 层零动态反射

核心程序集不依赖任何运行时代码生成，从源头保证 AOT 可裁剪

JIT-only 功能显式分离

NativeDynamicHook 等 JIT 专属能力单独成层，不污染 AOT 主链路

不兼容即快速失败

AOT 环境下不支持的能力在启动期明确报错，而非运行时静默降级

CLI 与 Gateway 均为 NativeAOT 制品

交付物不依赖 .NET 运行时安装，单文件即可部署

技术指标

亚秒级

启动时间目标：Agent 服务可随需拉起、快速回收

3 平台

Desktop 捆绑包 Windows / macOS / Linux 全部 NativeAOT

预估准入

每轮请求先预估 Token 再准入，配合 TokenJuice 压缩控制成本上限

AOT 不是发布时的编译开关，而是从 Core 层开始贯彻全栈的架构约束。

08

五层架构：从用户交互到扩展生态，职责分层、依赖单向向下

边界与依赖

网关是唯一入口

所有外部流量经网关收口：认证、限流、审计在此统一执行

Core 不依赖上层

核心层处于依赖链最底端，保证 NativeAOT 可独立裁剪与验证

扩展层可插拔

MCP、技能、插件、数字员工均可独立装卸，不影响内核稳定性

分层的意义：安全收口在网关、性能基座在 Core、生态活力在扩展层。

09

核心组件：八个程序集各司其职，交付物从 CLI 到管理台全覆盖

组件	职责	所属层
OpenClaw.Core	零动态反射基座：配置、依赖注入与基础原语	核心层
OpenClaw.Agent	Agent 循环、Goal / Loop / Pulse 调度、工具与 Hook 链	运行时层
OpenClaw.Gateway	统一端点、认证授权、频道接入（NativeAOT 制品）	网关层
OpenClaw.Cli	命令行入口：安装、启动、诊断（NativeAOT 制品）	交付层
OpenClaw.Companion	Desktop 伴随端，Avalonia 跨平台	交互层
Dashboard / Client / Tui	Blazor WASM 管理台、客户端 SDK、Spectre.Console 终端	交互层

扩展组件（MCP App、技能包、数字员工、插件）在扩展层独立打包、按需装卸。

组件边界与架构分层一一对应：每一层都有明确的交付物与负责人。

10

Agent 循环：ReAct 范式驱动的推理 — 行动 — 观察闭环

01 **输入**

来自频道、网关或调度引擎的用户请求

02 **推理 Thought**

模型规划下一步：直接回复或调用工具

03 **行动 Action**

经 Hook 拦截链把关后执行工具调用

04 **观察**

工具结果回写上下文

观察结果回流，进入下一轮推理

会话即状态，而非线程

会话状态持久化，支持跨批次后台续推与启动恢复

并发与资源护栏

上限 64 个会话、空闲 30 分钟超时；每轮 Token 写入 4 个槽位记账

循环的每一轮都可观测、可拦截、可记账 —— 这是运行时而非脚本的区别。

11

Goal、Loop、Pulse 三种调度引擎，覆盖从长任务到心跳巡检的全部节奏

Goal • 目标引擎

长任务推进与终止

状态机驱动：idle → running → completed / blocked / failed

128K Token 预算硬上限

连续 3 次相同文本 hash 触发阻塞

续推上限 200 轮 / 360 分钟

后台并发上限 3 路

Loop • 循环引擎

周期性任务注入

cron 调度：基于 TickerQ 的周期任务注入

简写 5m 展开为 */5 * * * *

任务注入后复用 Agent 循环执行

适合报表、巡检、同步等固定节奏工作

Pulse • 心跳引擎

定期自检与静默巡检

默认 30 分钟心跳间隔

HEARTBEAT.md 清单驱动检查项

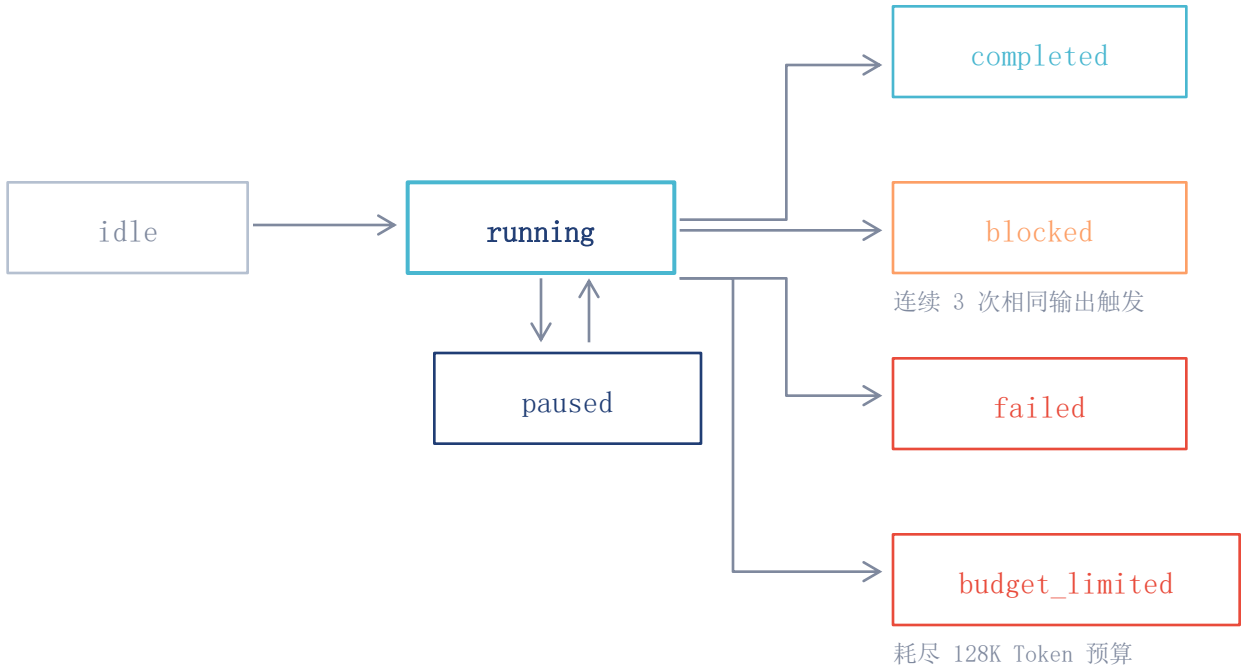
无异常时回复 HEARTBEAT_OK 保持静默

LightContext / IsolatedSession 轻量隔离执行

三引擎互补：Goal 推长任务、Loop 跑固定节奏、Pulse 做静默巡检。

12

Goal 引擎：状态机 + 多层预算护栏，保证长任务可终止、可暂停、可恢复



预算与护栏

128K Token 预算
耗尽即进入 budget_limited，拒绝失控燃烧

MaxContinuationTurns = 200
续推轮数硬上限，防止无限自续

MaxWallClockMinutes = 360
墙钟时间护栏，超时即终止

MaxIterationsPerBatch = 20
单批迭代上限，批量间可检查点

后台并发 3 路
MaxConcurrentBackgroundTurns 限制资源占用

护栏全部默认开启：长任务的预算、轮数、墙钟与并发都有硬边界。

13

所有工具调用必经 Hook 拦截链：审计、自治策略、契约治理依次把关

BridgedToolHook
桥接 Node.js 生态钩子，5 秒超时保护主循环

NativeDynamicHook
动态加载原生钩子，仅 JIT 模式可用（JIT-only）

故障安全
After 钩子异常自动故障安全；扁平列表全量执行

拦截链让"Agent 能做什么"从模型自觉变为运行时强制。

14 一个网关统一暴露聊天、管理、OpenAI 兼容、MCP、A2A 与集成端点

对话入口

/chat 网页聊天界面

/v1/chat/completions • **/v1/responses** OpenAI 兼容 API

/ws 实时流式通道

运维管理

/admin 管理界面

/health • **/doctor** 健康检查与诊断

协议互联

/mcp MCP 协议端点

/a2a • **/a2a/rpc** Agent-to-Agent 互联

Agent Card 发布于 /.well-known/agent-card.json，任务生命周期 submitted → working → completed / failed

开放集成

/api/integration/* 第三方系统集成

/webhooks/{name} 入站 Webhook

/apps/* 托管 MCP App 应用

所有流量经由同一网关：认证、限流与审计在此处统一收口。

15

六种认证方式 × 三种角色：从本机开发到企业 OIDC 的完整谱系

认证方式	适用场景	说明
loopback-open	本机开发	仅回环地址，开箱即用
bearer（Bootstrap Token）	脚本与自动化	启动引导令牌
操作员账户 + 密码	日常运维	PBKDF2 120K 迭代哈希
account_token	服务间调用	账户级访问令牌
browser-session	浏览器会话	Cookie 会话 + CSRF 防护
oidc_jwt	企业接入	对接组织 OIDC 身份

角 色 与 防 护

viewer / operator / admin
只读、运维、管理三级角色

组织策略
按组织维度下发权限与限制

CSRF + 速率限制
管理面默认开启跨站防护与请求限流

从 loopback 到 OIDC：同一份授权模型覆盖个人开发与企业部署。

16

默认安全、渐进硬化：Compatibility → Supervised → Lockdown

工具自主三级（与安全姿态联动）

full
完全自主，仅限受信环境

supervised（默认）
危险操作需批准后执行

readonly
只读工具集，可安全对外开放

安全不是一个开关，而是随环境成熟度逐级收紧的路径。

17

80+ 原生工具按 15 个域组织，从文件 Shell 到 Canvas 全面覆盖

文件与 Shell	6
记忆	4
Web	4
代码执行	3
通信	3
数据库	3
智能家居	4
日历与媒体	4
会话与委托	2
Canvas（A2UI）	14
Gateway 管理	10+
Goal 与 Loop	4
FractalMemory	7
MetaSkill	7
外部 MCP 工具	**按需接入**

数据来源：OpenClaw.NET 工具系统章节；外部 MCP 工具数量取决于接入的服务器。

工具按域注册、按自主级别授权，外部 MCP 工具经同一 Hook 链把关。

18

12 个频道统一接入：从 Telegram 到飞书、钉钉、企业微信

国际频道

Telegram WhatsApp Slack

Discord Microsoft Teams Signal

SMS Email Webhook

企业 IM

飞书 钉钉 企业微信

三个企业 IM 频道支持配置热重载，无需重启网关

严格 Allowlist 语义

未显式允许的发送者即拒绝，频道接入默认收敛

频道即适配器

消息归一化后进入同一 Agent 循环，人格跨频道一致

同一 Agent 人格在所有频道一致在线，接入面与安全语义同时收敛。

19

Model Profile：以能力而非厂商描述模型，让模型成为可替换资源

8 类模型提供商统一抽象

云端 API、本地推理、自托管服务收敛为同一接入面

提供商标识无关的能力描述层

Profile 描述"能做什么"，而非"是谁家的模型"

11 项 Capability 标志刻画能力

对话、流式、工具调用、视觉等能力维度结构化声明

FallbackProfileIds 回退链

主 Profile 不可用时沿回退链自动降级，服务不中断

预置 Profile

7 个

预置 Preset 开箱即用

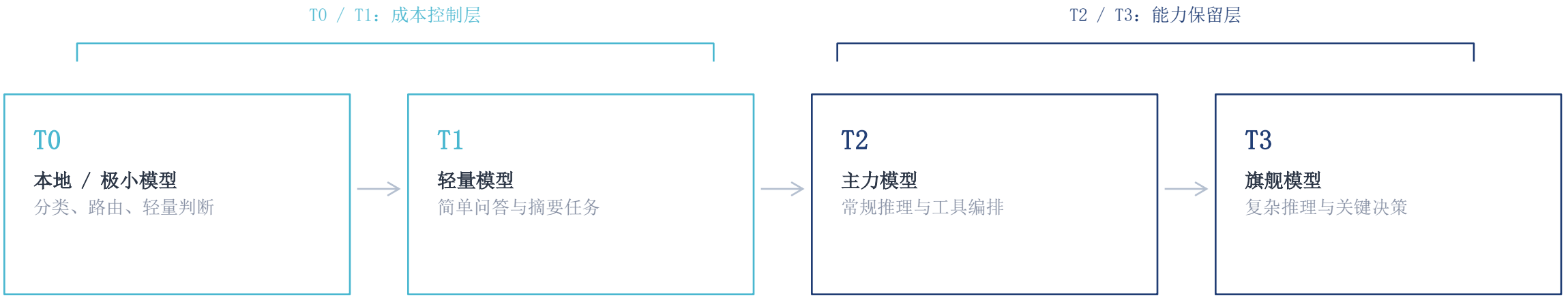
包含 **embedded Gemma 4 GGUF**：无外部依赖的本地嵌入推理，离线环境亦可运行。

换模型不改代码：路由与回退按能力匹配，与厂商解耦；本地与云端可混合编排。

Profile 把"选模型"从硬编码配置变成运行时的能力匹配问题。

20

T0 - T3 动态路由：简单请求走低成本层，复杂任务保留高能力层



与 Model Profile 双层协同

先按能力匹配 Profile，再按层级控制成本，两个维度独立演化

路由决策可观测

每次分层选择进入指标与审计，成本归因可复盘

分级路由把成本花在真正需要能力的请求上，而非平均摊给所有请求。

21

三层记忆架构：运行时记忆、Fractal Memory、Mempalace 各司其职

Layer 1 • 运行时记忆

file / sqlite（FTS5 全文检索为默认）：会话内事实与偏好，轻量随取随用

Layer 2 • Fractal Memory

MCP-first 的结构化项目记忆：跨会话沉淀项目知识与决策脉络

Layer 3 • Mempalace（可选）

时序知识图谱：实体、关系与时间维度的长期记忆，按需启用

治理与生命周期

Retention Sweeper

按保留策略定期清理过期记忆，记忆规模可控

写入可审计

记忆写入经审计链留痕，可追溯每条记忆的来源

默认轻量、按需增强

Layer 1 即可运行，Layer 2 / 3 按场景渐进启用

记忆分层：默认轻量、按需增强，保留策略本身也可被治理。

22

Harness 四层治理：从意图到执行，全程被动优先、审计完整

01 **Contract**
意图
声明要做什么、边界在哪里

→

02 **Evidence Bundle**
事实
收集证据与上下文，支撑判断

→

03 **Governance Ledger**
决策
决策留痕，可回放、可复盘

→

04 **PEV**
执行治理
批准后方能进入执行

四条治理原则

Passive First 被动优先
系统默认等待人工输入，不主动发起变更

永不自动生效
任何治理决策都必须经人批准后执行

审查优先于效率
宁可多一次人工确认，不少一道审查环节

审计完整
意图、证据、决策、执行全程留痕可回放

治理不是拦截器，而是一条从意图到执行的完整证据链。

23

技能系统双轨：提示词技能即刻可用，MetaSkill 以 DAG 编排 3－12 步

Standard 技能：提示词封装

1－2 步任务直接以提示词技能承载，兼容 SKILL.md 生态即刻复用

MetaSkill：DAG 编排 3－12 步

7 种步骤类型：llm_chat / llm_classify / agent / tool_call / skill_exec / user_input / fan_out；内存执行、Wave 并行调度无依赖步骤

13+ 步：移交 MAF Durable Workflow

外部持久化执行，RequestPort 支持异步人工审批断点续跑

工程约束与制品

on_failure 五项工程约束

每步失败策略显式声明，拒绝隐式重试

Jinja2 沙箱仅 4 个过滤器

模板能力最小化，堵住注入面

Ontology 投影

按信号评分选择 Topic 与 View，注入领域知识

Artifact 制品与 Stage Gate

emit_artifact 产出制品；多阶段契约关卡保证中间质量

1－2 步用技能、3－12 步用 MetaSkill、13+ 步用持久化工作流 —— 按复杂度选执行体。

MCP 三层：Server 向外、Client 向内、App 宿主化全生命周期管理

MCP Server：向外暴露

把 OpenClaw 的工具与资源以 MCP 协议暴露给外部客户端

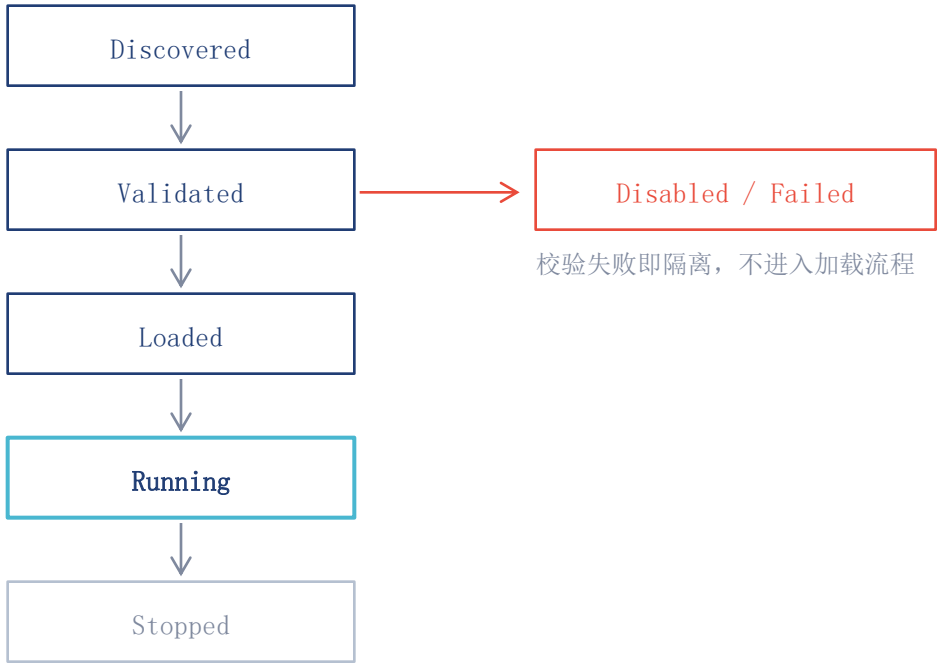
MCP Client：向内接入

接入第三方 MCP 服务器的工具，纳入同一 Hook 拦截链治理

MCP App：宿主化管理

以 openclaw.mcpapp.json 清单声明并托管完整 MCP 应用，47 个单元测试守护生命周期

MCP App 六阶段生命周期



MCP 不只是协议适配：应用在宿主内被发现、校验、加载、运行与退役。

TokenJuice：规则驱动的确定性压缩，零 LLM 成本换取 50–95% 压缩率

压缩发生在入上下文之前

工具输出先经 TokenJuice 压缩，再进入模型上下文窗口

规则驱动、确定性、零 LLM 成本

不调用模型做摘要，压缩本身不产生任何 Token 费用

多轮放大效应

上下文逐轮累积时，压缩收益随会话长度持续复利

与预估 Token 准入控制配合

先预估、再压缩、后准入，成本上限在入口处被锁死

压缩系数与压缩率

50 – 95%

α = 0.5
保留一半，压缩 50%

α = 0.05
保留 5%，压缩 95%

压缩系数 α ∈ [0.05, 0.5]，按工具输出类型与场景配置。

数据来源：OpenClaw.NET PRD v1.1 §TokenJuice。

压缩不调用模型：确定性规则在长会话中持续复利，成本优势随规模扩大。

26

六层可观测性：从启动标记到审计导出，58 个无锁计数器全程度量

01 **启动 Phase Markers**
启动各阶段耗时标记，冷启动慢在哪一目了然

02 **Doctor 双格式诊断**
人读与机读两种格式，体检结果可直接进自动化

03 **Posture 安全姿态**
当前安全配置的可视化报告，硬化进度可追踪

04 **Maintenance 扫描**
定期维护扫描，发现配置漂移与遗留风险

05 **Pulse 心跳**
运行时自我巡检，异常时主动上报

06 **事故与审计导出**
事故复盘材料与审计记录可一键导出

指标与审计底座

58 个

RuntimeMetrics 无锁计数器，高频路径零分配开销

三层审计 JSONL
结构化审计日志 + SHA256 哈希链防篡改

OpenTelemetry 集成
指标与链路可导出至既有可观测性平台

从启动到事故复盘，运行时的每一步都有据可查、有数可依。

27

数字员工：一个 ZIP 包交付身份、记忆与技能的完整人格

包内结构

AGENTS.md 行为准则与工作边界

SOUL.md 人格与语气定义

IDENTITY.md 姓名、角色等身份信息

MEMORY.md 初始长期记忆

skills/ 随包分发的技能目录

ontology 领域知识投影配置

安全与运维

30MB 包体上限 + ZIP-slip 三阶段验证
防止路径穿越与恶意包注入

技能热重载
更新技能无需重启运行时

OperatorAudit 操作审计
安装、更新、启停全程留痕

数字员工 = 可分发的完整人格：身份、记忆、技能，一个包交付、可审计地运行。

28

学习以审查为前提：观察 → 提案 → 审查 → 批准 → 生效

01 **观察**
系统记录交互模式

02 **提案**
生成 6 类变更提案

03 **审查**
人工核对证据与影响面

04 **批准**
显式授权后排队执行

05 **生效**
可回滚范围内落地

automation_suggestion 提案的质量管道

5 步质量管道

提案生成后逐级质检，低质量信号在到达用户前被拦截

四级质量决策

ready_draft（可提交）/ needs_review_draft（需修订）/ learning_only（仅学习）/ suppressed（抑制）

可回滚边界

可回滚范围明确

每类生效变更都声明回滚路径，试错成本有上限

与治理链同源

学习提案复用 Harness 审查与审计机制

系统可以学习，但任何变更都必须在人的审查之后生效。

29

三条部署路径、五层客户端：同一运行时从服务器到口袋设备

部署路径

源码检出：dotnet run

开发者本地最快路径，完整调试能力

Desktop 捆绑包：三平台全 NativeAOT

Windows / macOS / Linux 单文件分发，无需安装 .NET 运行时

Docker 容器

服务器标准化部署；可叠加 Tailscale Serve 安全暴露到内网之外

五层客户端

WebChat 浏览器聊天界面

Admin Dashboard Blazor WASM，16 个管理页面

Desktop Companion Avalonia 跨平台伴随端

TUI Spectre.Console 终端界面

Android AgentQi .NET MAUI 移动端

从 dotnet run 到 Docker：同一运行时在每种形态下保持一致能力。

30

对比四类竞品的 20 个维度后，差异化收敛到三条架构级判断

01 **NativeAOT 友好的架构约束已落到 Core 层**

零动态反射基座 + JIT-only 显式隔离 + 快速失败：AOT 不是事后适配，而是设计前提

02 **安全与治理是默认路径，而非可选插件**

Hook 拦截链、三级自主、Harness 被动治理与完整审计链全部内建于运行时

03 **成本优化内建于运行时：分级路由 + 零成本压缩**

T0 - T3 动态路由与 TokenJuice（50 - 95% 压缩率）让 Token 成本随规模可控

开放边界（如实标注）

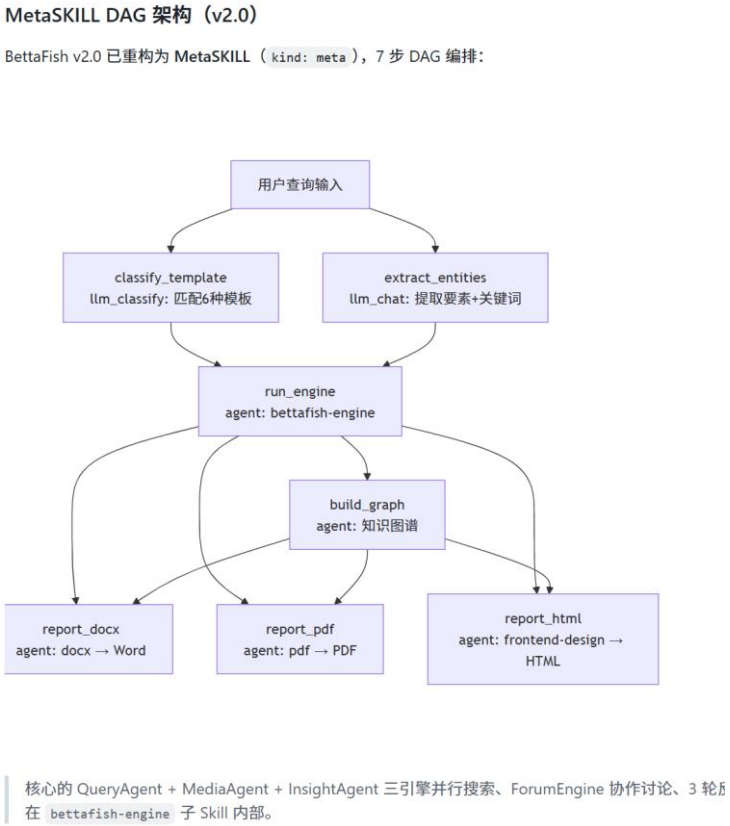
- 启动耗时、内存占用与压缩率的量化基准（benchmark）数据待补充；
- 与 OpenClaw / OpenSquilla / Hermes Agent / Claude Code 的 20 维逐项对比。

内容来源：OpenClaw.NET 产品需求文档 v1.1（2026-07-21）

OpenClaw.NET：让 .NET 生态拥有自己的 Agent 基础设施 —— 现在即可从源码开始验证。

31 Demo 演示

多智能体舆情分析 MetaSKILL —— 7 步 DAG 编排（classify → extract → engine → graph → 3×report），基于 QueryAgent + MediaAgent + InsightAgent 三引擎并行架构，通过 ForumEngine 实现 Agent 间协作讨论，生成 Word/PDF + 精美 HTML 双格式报



https://github.com/geffzhang/BettaFish-skill