

## 专业技能

- 云原生与 DevOps: Azure (Container, Functions, Serverless), Kubernetes (Helm, KEDA), Docker 多阶段构建, GitHub Actions (CI/CD), Google Cloud Platform (Cloud Run), 私有云/混合云部署架构, 零停机部署策略
- 后端架构: 微服务架构, 分布式系统设计, OAuth2.0/OIDC 统一认证网关, Python (FastAPI, asyncio), TypeScript (Node.js/Express), gRPC, Redis 缓存策略, PostgreSQL
- AI 基础设施与协议: Model Context Protocol (MCP) 核心开发, OpenAPI/Swagger 解析与转换引擎 (Strata), AI Agent 工具链集成, LLM 上下文管理, BM25+ 向量检索算法
- 系统编程与底层: Rust, C/C++, GNU/Linux 发行版构建 (Gentoo), 内核模块调试, WebAssembly, 嵌入式开发
- 前端技术: Next.js (SSR), React, Astro, Material Design
- 编程语言: Kotlin, Java, Python3, C#, Rust, C, PowerShell, Bash Script, JavaScript, TypeScript, SQL
- 开发工具: Emacs, (Neo)Vim
- 证书: RISC-V Foundational Associate (RVFA), 红帽认证工程师 (RHCE) - 285/300, 红帽认证系统管理员 (RHCSA) - 满分

## 开源贡献和技能亮点

- Klavis AI 开源生态 (AI Agent & MCP 协议):
  1. **Code Owner**: 累计提交 349 次, 贡献代码 165 万 + 行。主导 Open-Strata 开源项目, 将企业级 OpenAPI 转换引擎回馈社区
  2. [PR#788](#): 实现基于进程隔离的 Playwright MCP Server, 解决多租户浏览器自动化的安全与资源隔离问题
  3. [PR#833](#) / [PR#836](#): 扩展 MCP 生态, 集成 Azure AD、Sentry、HuggingFace 等 10+ 服务的 OAuth 认证与工具定义
- 个人开源项目:
  1. [UpgradeAll \(1k+★\)](#): 跨平台应用更新管理器。采用 Kotlin (Android) + Rust (Core) 架构, 设计了模块化的更新源插件系统
  2. [distrobox-boost](#) / [numlockw](#): Rust 容器运行时优化工具与 Wayland 协议底层输入控制工具
- 底层系统与社区贡献:
  1. **Gentoo Linux**: 维护者, 贡献了 sys-fs/zfs (内核实验特性 [PR#44885](#))、kde-plasma 等关键包的修补与维护
  2. [CachyOS-kernels](#): CachyOS 优化内核 Gentoo overlay 维护者, 第一时间跟进上游内核更新并维护补丁兼容性
  3. **virtio-win** (Windows 半虚拟化驱动):
    - viogpu 驱动: 实现动态帧缓冲区大小调整, 支持 8K+ 分辨率的间接描述符 ([PR#1479](#)), 修复错误路径中的蓝屏崩溃 ([PR#1473](#)) 及初始化失败时的资源泄漏 ([PR#1475](#))
    - 安装程序: 修复驱动占用时升级失败的问题, 避免系统升级时出现 error 1603 ([PR#85](#))
  4. **GradleAndroidRustPlugin**: 解决 Rust/Android 交叉编译时的 ABI 兼容性与 Gradle 9 升级适配 ([PR#10](#), [PR#11](#))
  5. **InfiniTime**: 为智能手表固件贡献 CMake 构建修复与中文支持 ([PR#2143](#))
- 曾入选 Linux 基金会人才激励计划, 适应跨时区合作的工作

## 工作经历

### Klavis AI - 全栈工程师

2025.7 - 2025.12

作为核心创始工程师，主导 AI Agent 集成平台的基础设施建设。负责从底层协议实现 (MCP) 到上层 OpenAPI 转换引擎 (Strata) 的全栈架构设计，解决了 AI 模型与外部 20+ SaaS 服务互操作的标准化难题。

核心架构与平台建设：

- 主导研发 Strata OpenAPI 引擎：设计核心解析引擎，能够将任意 OpenAPI v2/v3 规范自动转换为标准化的 MCP 工具定义
- 极致性能优化：通过 OpenAPI 对象预构建缓存和 jsonref 深度解析优化，将复杂规范的解析加载时间从 15 分钟（单线程）缩短至 3 秒，性能提升 300 倍
- 生产环境价值：彻底解决了 Google Cloud Run 在 Serverless 冷启动和高并发扩容时的性能瓶颈，实现了零停机部署
- 设计企业级 MCP 运行时架构：基于进程隔离模型确保 Playwright 等高风险工具在独立沙箱中运行；构建支持 Microsoft Entra (Azure AD)、GitHub、PayPal、Sentry 等 10+ 平台的统一 OAuth 认证网关

DevOps 与私有云交付：

- 从零构建私有化部署架构：设计并实施基于 Kubernetes + Helm 的完整私有云交付方案，实现 MCP 服务的按需加载与自动休眠机制
- 弹性伸缩：集成 KEDA HTTP Add-on，实现基于实时 HTTP 流量的 Pod 自动扩缩容
- 全栈 CI/CD 流水线：构建 GitHub Actions 自动化体系，实现 Docker 镜像的多架构 (amd64/arm64) 构建与自动分发

### 微软 (外包) - Office365 互操作性和合规项目 - 全栈工程师

2023.10 - 2025.7

负责 Office 文档生态系统的工具链开发与合规性工程，主导了内部系统的云原生转型与自动化测试体系建设。

- 云原生架构转型：将遗留的.NET Framework 单体应用解耦为基于 Azure Container 和 Azure Functions 的微服务架构，利用 Azure Managed Identity 替代传统凭据，将资源成本降低 85% 并消除了安全合规风险
- 自动化测试平台：设计“代码即配置”的 Python 自动化测试框架，结合 OCR 视觉识别技术和 GPT 模型，实现了 Office Word 字符渲染兼容性的端到端自动化测试，将人工测试工作量减少 80%
- 文档工程智能化：深入研究 Office Open XML (OOXML) 底层结构，开发基于 LLM 的文档术语一致性校验工具，将 GB 认证文档的修正效率提升 96%（从 2.5 小时缩短至 5 分钟）

### 拓维信息 - 联通 A 股门户网站 - JAVA 后端开发

2022.2 - 2022.7

联通为宣传 A 股的门户网站，因年久失修且技术落后，难以部署与维护网站内容。负责项目后端的微服务设计和代码重写，并实现了 k8s 的流水线部署与自动化扩/缩容。

- 使用微服务架构进行项目重构
- 采用敏捷开发方法开发
- 实现了可动态拓展的 k8s 部署

## 项目经历

### Klavis AI - Strata OpenAPI 集成平台 - Python 开发

2025.8 - 2025.12

项目背景：AI Agent 需要调用各种外部 API，但每个 API 都有不同的规范和认证方式。Strata 平台旨在将任意 OpenAPI 规范自动转换为 MCP 工具，实现 AI Agent 对 API 的无缝调用。

项目贡献：

- 从零搭建项目架构，设计可扩展的服务注册机制，实现 OpenAPI v2/v3 规范的解析与自动转换
- 实现 BM25+ 搜索算法用于 API 文档检索，支持标签过滤和统一搜索接口
- 设计懒加载架构实现 MCP 服务器按需加载，显著降低内存占用和启动时间
- 实现 tool\_lists 缓存机制和 OpenAPI 对象预构建缓存，CI 阶段预热加速生产环境启动

项目成果：

- 成功集成 Cloudflare、GitLab、Vercel、GitHub、Discord、PayPal、Sentry 等 20+ 个 SaaS 服务

<ul style="list-style-type: none"> <li>开源发布 Open-Strata 1.0.2, 将企业级 OpenAPI 转换引擎回馈社区</li> </ul> <p><b>Klavis AI - MCP Server 开发与认证集成 - Python 开发</b></p> <p>项目背景: Model Context Protocol (MCP) 是 AI Agent 与外部工具交互的标准协议。为扩展 Klavis 平台的工具生态, 需要开发新的 MCP Server 并构建统一的 OAuth 认证网关。</p> <p>项目贡献:</p> <ul style="list-style-type: none"> <li>独立开发 4 个 MCP Server: Dropbox (文件管理)、QuickBooks (财务 CRUD)、Microsoft Office 365 (Teams/Outlook/OneDrive)、Playwright (浏览器自动化), 并优化多个现有服务</li> <li>设计进程隔离架构: Playwright Server 采用独立沙箱运行, 解决多租户环境下的安全与资源隔离问题</li> <li>构建统一 OAuth 认证网关: 完整实现 Microsoft Entra (Azure AD) 企业级认证, 集成 Discord、GitLab、Vercel、Shopify 等 10+ 平台</li> <li>设计 Tool Call Logs 系统: 实现用户级别 API 调用追踪与后台异步日志写入, 实现 AI Agent 与 MCP Tools 调用的可调试性</li> </ul>	2025.7 - 2025.12
<p><b>Klavis AI - MCP Sandbox 测试框架 - Python 开发</b></p> <p>项目背景: MCP 集成涉及多个外部服务, 需要沙盒测试环境来验证功能正确性并保证集成质量。</p> <p>项目贡献:</p> <ul style="list-style-type: none"> <li>设计并实现 7 个沙盒测试环境: 覆盖 Cal.com、QuickBooks、Dropbox、Shopify 等服务的完整功能测试</li> <li>建立严格内容验证机制与 Token 刷新逻辑测试, 实现 MCP 集成的自动化质量保障</li> </ul>	2025.9 - 2025.12
<p><b>Klavis AI - 私有云部署方案 - DevOps 开发</b></p> <p>项目背景: 企业客户需要在自有基础设施上部署 Klavis 平台, 需要设计完整的私有云/On-Premise 部署方案。</p> <p>项目贡献:</p> <ul style="list-style-type: none"> <li>从零搭建 Kubernetes + Helm 部署架构: 完整的 Helm Chart 配置、服务编排和依赖管理</li> <li>集成 KEDA HTTP Add-on: 实现基于 HTTP 流量的 Pod 自动扩缩容</li> <li>设计 ClusterIP + Endpoints 同步方案替代 ExternalName, 优化超时配置</li> <li>编写中英文 README、故障排查指南和流量路径图</li> </ul> <p>项目成果:</p> <ul style="list-style-type: none"> <li>实现可扩展的企业级私有云部署架构</li> <li>支持 MCP 服务的按需加载与自动休眠, 优化资源利用率</li> </ul>	2025.11 - 2025.12
<p><b>开源项目 - UpgradeAll 全栈应用更新器 - 项目发起人</b></p> <p>项目链接: <a href="#">UpgradeAll</a> (Kotlin/Rust 客户端), <a href="#">Server</a> (Python 服务端, 2020.3-2022.6.5)</p> <p>领导六人团队进行协作开发免费开源软件 UpgradeAll, 解决传统更新存在的软件发布碎片化的问题。简化 Android 应用 (包括未安装的应用)、Magisk 模块等的更新查找过程。项目致力于提供高速且易用的应用更新体验。客户端获得 1k+Star</p> <p>客户端亮点 (Kotlin+Rust):</p> <ul style="list-style-type: none"> <li>采用 Kotlin 开发前端, 实现 Material Design 界面及相关组件</li> <li>使用 Rust 开发高性能后端库, 采用模块化代码设计, 内核可独立使用</li> <li>实现高度可自定义设置, 支持通过 Json 配置更新来源。内嵌 JavaScript 引擎实现应用热更新能力</li> </ul> <p>服务端亮点 (Python):</p> <ul style="list-style-type: none"> <li>提供客户端 gRPC 和 REST 接口, 支持从 GitHub、GitLab、F-Droid、Play Store 等多个源获取应用更新</li> <li>使用 ZeroMQ 实现微服务架构与服务发现, 设计可横向扩展的多层缓存服务架构</li> <li>采用 Redis 实现分布式数据缓存, 使用 Docker 容器化技术部署服务</li> </ul> <p>项目成果:</p> <ul style="list-style-type: none"> <li>将 30 分钟更新时间缩短至 2 分钟显著缩短用户应用更新时间, 将原本需要手动查找的过程自动化</li> <li>构建一站式应用更新平台, 集成多个更新源</li> </ul>	2019.4 - 至今

## 微软 - Office Word 自动化测试平台 - Python 开发

2024.3 - 2025.7

项目背景：Office Word 需要大量字符兼容性和渲染测试来确保产品质量，传统人工测试方法效率低下且占用测试团队大量资源。为解决这一问题，我负责开发自动化测试平台，实现每周数百个测试用例的自动执行。

项目贡献：

- 设计并实现 Python 自动化测试框架，专用于 Office Word 软件的兼容性验证
- 创新性采用“代码即配置”的测试系统架构，使测试人员能通过 AI 辅助和 IDE 环境高效定义测试规则
- 集成 Windows API 接口模拟用户操作，构建端到端的自动化测试流程
- 开发基于 OCR 技术的渲染结果识别系统，提高字符兼容性测试的准确性
- 实现虚拟机环境中的批量测试调度功能，支持大规模并行测试执行

项目成果：

- 实现每周自动执行 800+ 测试用例，测试覆盖率显著提升。将测试团队投入的人工测试时间减少约 50%
- 通过系统化测试提高产品质量，降低字符渲染相关问题的用户反馈率

## 微软 - GB 文档校验与修正工具 - Python 开发

2024.6 - 2025.7

项目背景：微软 Office 套件需通过 GB 认证，要求测试团队提供标准化的测试结果文档。但合并文档时，常面临术语不统一和格式错乱等问题。为解决这些问题，我开发自动化工具对文档进行校验与修正，确保提交的文档符合标准要求。

项目贡献：

- 深入研究 Word 文档 Open XML 内部结构，掌握文档格式控制机制
- 实现 python-docx 未支持的剪切、复制和粘贴功能
- 利用 LLM 实现专业术语智能替换功能，确保文档术语表达一致性
- 利用 python-docx 和 oxml 库进行 OXML 底层操作，解决了多文档合并后出现排版错误的问题

项目成果：

- 将文档校验与修正时间减少 96.67%，从人均 2.5 小时缩短至 5 分钟
- 确保所有 GB 认证文档的术语一致性和格式规范性，提高认证通过率
- 减少测试团队在文档格式调整上的工作量，使其专注于测试内容本身
- 编写技术博客“[Cut and move Runs via python-docx](#)”，为开源社区贡献解决方案

## 微软 - Gendox 文档管理系统 - C# 开发

2023.10 - 2025.7

项目背景：Gendox 是微软内部的文档管理工具，以 Word 插件形式自动转化为 wiki。为产品经理提供结构化文档编写环境，采用“先建菜单再填内容”的方法，支持文档片段跨文档共享与同步修改。该项目涵盖从编辑到发布的全流程，集成了版本控制、自动化构建和安全保障等核心功能。

项目贡献：

- 调研 GenDox 插件加载与运行效率，开发基于 Python 和图像识别的自动化测试工具
- 基于 Azure Pipeline 构建自动发布系统，实现新版本的持续交付。并重构 Release 工具
- 设计 Azure Function 自动归档方案，集成 PowerBI 自动化遥测数据采集生成实时看板与邮件预警系统
- 升级安全模型，将基于密码的认证迁移至 Azure Managed Identity，并编写标准化迁移文档
- 开发基于 Azure Serverless 的自动化工具，实现 VM 的 Patch Tuesday 更新自动应用

项目成果：

- 高效处理每周高达 30G、40 万文件的 Release 文件
- 简化团队协作流程，减少新版本测试时间，消除跨团队人工交接，节省每次发布约 3 人/天的工作量
- 实现日志自动检查，消除人工审查可能造成的遗漏风险，减少潜在延误和损失
- 自动化月度维护工作，节省每月 1 人/天的系统检查与更新时间
- 完成微软 Q3 季度安全要求，提升系统整体安全性

## 微软 - Interop 部门数据同步与培训管理系统改进 - C# 开发

2023.10 - 2025.7

项目背景：该系统作为 Office Interop 部门的核心工具，承担跨项目人员和文档的数据仓库同步，和员工培训管理两大关键职能。旧系统存在运行缓慢，技术与安全架构落后等问题。

项目贡献：

- 原有基于 Task Scheduler 的固定时间执行模式每天运行超过 12 小时，用 Power Shell 实现服务依赖脚本和碎片化执行
- 升级项目安全架构以满足最新安全标准，将 CodeQL 集成至 Azure Pipeline，实现代码安全的自动化检测与持续集成
- 领导项目微服务化转型，将单体应用解耦为独立服务组件，通过 Azure Container 技术实现从.NET Framework 向.NET

项目成果：

- 将 Azure 资源费用降低 85%，调度系统将运行时间降低 50%，提升吞吐量等和稳定性
- 全面达成微软最新安全合规要求，微服务架构彻底消除 VM 维护相关的安全风险
- 微服务的灰度迁移方案实现系统零中断升级，保障用户体验持续平稳流畅

## 清华大学实验室项目 - 基于 IPFS 的文件分享应用 - Android 客户端开发

2021.4 - 2021.5

项目背景：传统的文件传输方式存在带宽限制、服务器依赖性高等问题。该项目旨在利用 IPFS (星际文件系统) 的分布式特性，构建一个同时支持面对面高速传输和远距离稳定共享的文件分享应用。

指导老师：赵黎

项目贡献：

- 设计并开发 Android 客户端原型，实现核心功能和用户界面
- 集成 IPFS 协议，构建高效的 P2P 文件传输网络
- 实现基于 Wi-Fi Direct 的面对面传输功能，大幅提高近距离传输速度
- 开发端到端加密系统，确保文件传输安全性
- 设计直观的文件预览界面，优化用户体验

项目成果：

- 近距离传输速度达到传统云存储解决方案的 3-5 倍，达到 1GB/S
- 成功实现不依赖中心服务器的 P2P 文件分享系统，提高性能和稳定性。
- 作为研究生课题的核心实现部分，获得指导老师高度评价

## 教育经历

### 华北理工大学 - 计算机科学与技术 - 本科

2023.6

课程：网络原理，计算机原理，软件工程，算法设计与分析，面向对象程序设计，数据库原理，操作系统（助教）

## 获奖经历

### ASC18 世界大学生超级计算机竞赛 - 二等奖

## Professional Skills

---

- Cloud Native & DevOps: Azure (Container, Functions, Serverless), Kubernetes (Helm, KEDA), Docker multi-stage builds, GitHub Actions (CI/CD), Google Cloud Platform (Cloud Run), Private/hybrid cloud deployment architecture, Zero-downtime deployment strategies
- Backend Architecture: Microservice architecture, Distributed systems design, OAuth2.0/OIDC unified authentication gateway, Python (FastAPI, asyncio), TypeScript (Node.js/Express), gRPC, Redis caching strategies, PostgreSQL
- AI Infrastructure & Protocols: Model Context Protocol (MCP) core development, OpenAPI/Swagger parsing and conversion engine (Strata), AI Agent toolchain integration, LLM context management, BM25+ vector retrieval algorithms
- Systems Programming & Low-level: Rust, C/C++, GNU/Linux distribution building (Gentoo), Kernel module debugging, WebAssembly, Embedded development
- Frontend Technology: Next.js (SSR), React, Astro, Material Design
- Programming Languages: Kotlin, Java, Python3, C#, Rust, C, PowerShell, Bash Script, JavaScript, TypeScript, SQL
- Development Tools: Emacs, (Neo)Vim
- Certifications: RISC-V Foundational Associate (RVFA), Red Hat Certified Engineer (RHCE) - 285/300, Red Hat Certified System Administrator (RHCSA) - Perfect Score

## Open Source Contributions and Key Skills

---

- Klavis AI Open Source Ecosystem (AI Agent & MCP Protocol):
  1. Code Owner: 349 commits, contributed 1.65M+ lines of code. Led the Open-Strata open source project, contributing the enterprise-grade OpenAPI conversion engine back to the community
  2. [PR#788](#): Implemented process-isolated Playwright MCP Server, solving security and resource isolation issues for multi-tenant browser automation
  3. [PR#833](#) / [PR#836](#): Extended MCP ecosystem, integrating OAuth authentication and tool definitions for 10+ services including Azure AD, Sentry, HuggingFace
- Personal Open Source Projects:
  1. [UpgradeAll \(1k+★\)](#): Cross-platform application update manager. Built with Kotlin (Android) + Rust (Core) architecture, designed modular update source plugin system
  2. [distrobox-boost](#) / [numlockw](#): Rust container runtime optimization tool and Wayland protocol low-level input control tool
- Low-level Systems & Community Contributions:
  1. Gentoo Linux: Maintainer, contributed patches for sys-fs/zfs (kernel experimental features [PR#44885](#)), kde-plasma, and other critical packages
  2. [CachyOS-kernels](#): Maintainer of Gentoo overlay for CachyOS optimized kernels, actively tracking upstream kernel releases and maintaining patch compatibility
  3. virtio-win (Windows Paravirtualization Drivers):
    - viogpu Driver: Implemented dynamic framebuffer resizing with indirect descriptor support for 8K+ resolutions ([PR#1479](#)), fixed BSOD crash in error path ([PR#1473](#)), and resource leak on init failure ([PR#1475](#))
    - Installer: Fixed driver upgrade failure when drivers are in use, preventing error 1603 during system upgrades ([PR#85](#))
  4. GradleAndroidRustPlugin: Resolved Rust/Android cross-compilation ABI compatibility and Gradle 9 upgrade adaptation ([PR#10](#), [PR#11](#))
  5. InfiniTime: Contributed CMake build fixes and Chinese support for smartwatch firmware ([PR#2143](#))
- Selected for Linux Foundation Talent Incentive Program, adaptable to cross-timezone collaboration

# Work Experience

## Klavis AI - Full Stack Engineer / MCP Platform Architect

2025.7 - 2025.12

As a core founding engineer, led the infrastructure development for the AI Agent integration platform. Responsible for full-stack architecture design from low-level protocol implementation (MCP) to the upper-layer OpenAPI conversion engine (Strata), solving the standardization challenges of AI model interoperability with 20+ external SaaS services.

### Core Architecture & Platform Development:

- Led Strata OpenAPI Engine Development: Designed the core parsing engine capable of automatically converting any OpenAPI v2/v3 specification into standardized MCP tool definitions
- Extreme Performance Optimization: Through OpenAPI object pre-build caching and jsonref deep parsing optimization, reduced complex specification parsing time from 15 minutes (single-threaded) to 3 seconds, a 300x performance improvement
- Production Value: Completely resolved Google Cloud Run performance bottlenecks during Serverless cold starts and high-concurrency scaling, achieving zero-downtime deployment
- Enterprise MCP Runtime Architecture: Designed process isolation model ensuring high-risk tools like Playwright run in independent sandboxes; built unified OAuth authentication gateway supporting 10+ platforms including Microsoft Entra (Azure AD), GitHub, PayPal, Sentry

### DevOps & Private Cloud Delivery:

- Built On-Premise Deployment Architecture from Scratch: Designed and implemented complete private cloud delivery solution based on Kubernetes + Helm, implementing on-demand loading and auto-hibernation mechanisms for MCP services
- Elastic Scaling: Integrated KEDA HTTP Add-on, implementing Pod auto-scaling based on real-time HTTP traffic
- Full-Stack CI/CD Pipeline: Built GitHub Actions automation system, implementing multi-architecture (amd64/arm64) Docker image builds, security scanning (Trivy), and automatic distribution

## Microsoft (Vendor) - Office365 Interoperability and Compliance Project - Full Stack Engineer

2023.10 - 2025.7

Responsible for toolchain development and compliance engineering for the Office document ecosystem, led cloud-native transformation of internal systems and automated testing infrastructure development.

- Cloud-Native Architecture Transformation: Decoupled legacy .NET Framework monolithic applications into microservice architecture based on Azure Container and Azure Functions, replaced traditional credentials with Azure Managed Identity, reducing resource costs by 85% and eliminating security compliance risks
- Automated Testing Platform: Designed "code as configuration" Python automation testing framework, combined OCR visual recognition technology and GPT models, implemented end-to-end automated testing for Office Word character rendering compatibility, reducing manual testing workload by 80%
- Document Engineering Intelligence: Deep research into Office Open XML (OOXML) underlying structure, developed LLM-based document terminology consistency verification tool, improving GB certification document correction efficiency by 96% (from 2.5 hours to 5 minutes)

## Talkweb Information - China Unicom A-Share Portal Website - JAVA Backend Developer

2022.2 - 2022.7

The Unicom A-Share portal website, due to years of neglect and outdated technology, was difficult to deploy and maintain. Responsible for the microservice design and code rewrite of the project backend, implementing k8s pipeline deployment and automated scaling.

- Restructured the project using microservice architecture
- Adopted agile development methods
- Implemented dynamically scalable k8s deployment

# Project Experience

**Open Source Project** - UpgradeAll Full Stack Application Updater - Project Initiator 2019.4 - Present

Project Links: [UpgradeAll](#) (Kotlin/Rust client), [Server](#) (Python server, 2020.3-2022.6.5)

Led a six-person team in collaborative development of the free open-source software UpgradeAll, solving the fragmentation problem in traditional software updates. Simplified the update search process for Android applications (including uninstalled apps), Magisk modules, etc. The project aims to provide a fast and user-friendly application update experience. The client has received 1k+ Stars.

Client Highlights (Kotlin+Rust):

- Developed frontend with Kotlin, implementing Material Design interface and related components
- Developed high-performance backend library using Rust, with modular code design; the kernel can be used independently
- Implemented highly customizable settings, supporting update sources configuration via JSON. Embedded JavaScript engine enabling application hot update capability

Server Highlights (Python):

- Provided client gRPC and REST interfaces, supporting application updates from multiple sources including GitHub, GitLab, F-Droid, Play Store, etc.
- Used ZeroMQ to implement microservice architecture and service discovery, designed a horizontally scalable multi-layer cache service architecture
- Used Redis to implement distributed data caching, deployed services using Docker containerization technology

Project Outcomes:

- Significantly reduced user application update time from 30 minutes to 2 minutes, automating what was previously a manual search process
- Built a one-stop application update platform integrating multiple update sources

**Microsoft** - Office Word Automated Testing Platform - Python Development

2024.3 - 2025.7

Project Background: Office Word requires extensive character compatibility and rendering tests to ensure product quality. Traditional manual testing methods are inefficient and consume significant resources from the testing team. To address this issue, I was responsible for developing an automated testing platform to execute hundreds of test cases weekly.

Project Contributions:

- Designed and implemented a Python automation testing framework specifically for Office Word software compatibility verification
- Innovatively adopted a "code as configuration" test system architecture, allowing testers to efficiently define test rules through AI assistance and IDE environment
- Integrated Windows API interfaces to simulate user operations, building end-to-end automated testing processes
- Developed an OCR-based rendering result recognition system to improve the accuracy of character compatibility testing
- Implemented batch test scheduling in virtual machine environments to support large-scale parallel test execution

Project Outcomes:

- Achieved automated execution of 800+ test cases weekly, significantly improving test coverage. Reduced manual testing time by approximately 50%
- Improved product quality through systematic testing, reducing user feedback rates for character rendering issues

**Microsoft** - GB Document Verification and Correction Tool - Python Development

2024.6 - 2025.7

Project Background: Microsoft Office suite needs to pass GB certification, requiring standardized test result documents from the testing team. When merging documents, issues such as inconsistent terminology and format disruption often occur. To solve these problems, I developed an automated tool to verify and correct documents, ensuring they meet standard requirements.

Project Contributions:

- Thoroughly researched the internal structure of Word document Open XML, mastering document format control mechanisms
- Implemented cut, copy, and paste functionalities not supported by python-docx
- Utilized LLM for intelligent replacement of professional terminology, ensuring document terminology consistency
- Used python-docx and oxml libraries for OXML low-level operations, solving formatting errors that occurred after merging multiple documents

#### Project Outcomes:

- Reduced document verification and correction time by 96.67%, from an average of 2.5 hours to 5 minutes per person
- Ensured terminology consistency and format standardization for all GB certification documents, improving certification pass rates
- Reduced the testing team's workload on document formatting adjustments, allowing them to focus on the test content itself
- Wrote a technical blog "[Cut and move Runs via python-docx](#)", contributing solutions to the open source community

### **Microsoft** - Gendox Document Management System - C# Development

2023.10 - 2025.7

**Project Background:** Gendox is Microsoft's internal document management tool, which automatically converts to wiki in the form of a Word plugin. It provides product managers with a structured document writing environment, adopting the "build menu first, then fill in content" method, supporting cross-document sharing and synchronous modification of document fragments. The project covers the entire process from editing to publishing, integrating core functions such as version control, automated building, and security assurance.

#### Project Contributions:

- Researched GenDox plugin loading and running efficiency, developed Python-based and image recognition-based automated testing tools
- Built an automatic publishing system based on Azure Pipeline, achieving continuous delivery of new versions. Refactored the Release tool
- Designed an Azure Function automatic archiving solution, integrated PowerBI automated telemetry data collection to generate real-time dashboards and email alert systems
- Upgraded the security model, migrating from password-based authentication to Azure Managed Identity, and wrote standardized migration documentation
- Developed Azure Serverless-based automation tools to automatically apply Patch Tuesday updates for VMs

#### Project Outcomes:

- Efficiently processed up to 30GB and 400,000 files weekly for releases
- Simplified team collaboration processes, reduced new version testing time, eliminated cross-team manual handovers, saving approximately 3 person-days of work per release
- Implemented automatic log checking, eliminating oversight risks from manual reviews, reducing potential delays and losses
- Automated monthly maintenance work, saving 1 person-day per month for system inspection and updates
- Completed Microsoft Q3 quarter security requirements, enhancing overall system security

### **Microsoft** - Interop Department Data Synchronization and Training Management System Improvement - C# Development

2023.10 - 2025.7

**Project Background:** This system serves as a core tool for the Office Interop department, undertaking two key functions: cross-project personnel and document data warehouse synchronization, and employee training management. The old system had issues such as slow operation, outdated technology, and security architecture.

#### Project Contributions:

- The original Task Scheduler-based fixed-time execution mode ran for over 12 hours daily; implemented service dependency scripts and fragmented execution with PowerShell

- Upgraded the project security architecture to meet the latest security standards, integrated CodeQL into Azure Pipeline to achieve automated detection and continuous integration of code security
- Led the project's microservice transformation, decoupling the monolithic application into independent service components, implementing migration from .NET Framework to .NET through Azure Container technology

#### Project Outcomes:

- Reduced Azure resource costs by 85%, decreased scheduling system runtime by 50%, improved throughput and stability
- Fully met Microsoft's latest security compliance requirements; microservice architecture completely eliminated VM maintenance-related security risks
- The microservice gray migration solution achieved zero-downtime system upgrades, ensuring a continuously smooth user experience

### **Tsinghua University Laboratory Project** - IPFS-based File Sharing Application - Android Client Development 2021.4 - 2021.5

Project Background: Traditional file transfer methods face bandwidth limitations and high server dependency. This project aimed to leverage the distributed nature of IPFS (InterPlanetary File System) to build a file sharing application that supports both face-to-face high-speed transmission and stable long-distance sharing.

Supervisor: Li Zhao

#### Project Contributions:

- Designed and developed Android client prototype, implementing core functionalities and user interface
- Integrated IPFS protocol, building an efficient P2P file transfer network
- Implemented Wi-Fi Direct-based face-to-face transfer functionality, significantly improving short-distance transfer speed
- Developed end-to-end encryption system, ensuring file transfer security
- Designed intuitive file preview interface, optimizing user experience

#### Project Outcomes:

- Short-distance transfer speed reached 3-5 times that of traditional cloud storage solutions, achieving 1GB/S
- Successfully implemented a P2P file sharing system without relying on central servers, improving performance and stability
- Served as the core implementation part of a graduate thesis, highly praised by the supervisor

## **Education**

**North China University of Science and Technology** - Computer Science and Technology - Bachelor's Degree 2023.6  
Courses: Network Principles, Computer Principles, Software Engineering, Algorithm Design and Analysis, Object-oriented Programming, Database Principles, Operating Systems (Teaching Assistant)

## **Awards & Honors**

**ASC18 World University Supercomputer Competition** - Second Prize