DeerFlow 2.0: агент-харнесс с песочницей, памятью и субагентами

DeerFlow — открытый проект ByteDance, который вырос из инструмента для Deep Research в полноценный super agent harness: систему, дающую LLM-агенту песочницу, долгосрочную память и умение порождать субагентов для задач, растянутых на минуты и часы. Построен поверх LangChain и LangGraph. Ниже — разбор архитектуры, ключевых механизмов и того, как развернуть проект у себя.

> **Важно:** по умолчанию DeerFlow рассчитан на локальное доверенное окружение (доступ только через 127.0.0.1). Агент умеет выполнять системные команды и работать с файлами — если разворачиваешь его на сервере с сетевым доступом, обязательно настрой IP-allowlist и аутентификацию на реверс-прокси, иначе высокопривилегированные функции окажутся доступны извне.

1. Что такое DeerFlow 2.0

- **Расшифровка:** DeerFlow — Deep Exploration and Efficient Research Flow, харнесс, оркеструющий субагентов, память и песочницы.
- **История:** первая версия была фреймворком для Deep Research; 2.0 — полностью переписанный с нуля харнесс, не имеющий общего кода с v1 (старая ветка сохранена как `main-1.x`).
- **Лицензия:** MIT, открытый исходный код, репозиторий `bytedance/deer-flow`.
- **Звёзды на GitHub:** на момент проверки — около 68 тысяч. Счётчик звёзд меняется постоянно, поэтому цифры из разных источников и в разное время могут отличаться — ориентируйся на живой счётчик в самом репозитории.
- **Признание:** 28 февраля 2026 года DeerFlow занял 1-е место в GitHub Trending после релиза второй версии.

2. Субагенты и декомпозиция задач

- Ведущий агент (lead agent) на лету порождает субагентов — у каждого свой изолированный контекст, набор инструментов и условия завершения.
- Субагенты выполняются параллельно, где это возможно, и возвращают ведущему агенту структурированные результаты для синтеза в единый ответ.
- **Пример:** одна исследовательская задача может разойтись на десяток субагентов — каждый копает свой угол, а результаты сходятся в отчёт, сайт или презентацию со сгенерированной визуализацией.
- **Изоляция контекста:** субагент не видит контекст главного агента или других субагентов — это защищает его от отвлечения посторонней информацией и держит фокус на задаче.

3. Skills и инструменты

- Skill — структурированный markdown-модуль с описанием workflow, лучших практик и ссылок на вспомогательные ресурсы; встроены skills для research, генерации отчётов, слайдов, веб-страниц, изображений и видео.
- **Прогрузка:** skills подгружаются по требованию конкретной задачи, а не все сразу — это держит контекстное окно компактным даже на моделях с ограниченным контекстом.
- **Расширение:** можно добавлять свои skills, заменять встроенные или комбинировать их в составные workflow; кастомные skills лежат в `/mnt/skills/custom`.
- **Инструменты:** базовый набор — веб-поиск, веб-фетч, файловые операции, выполнение bash; плюс кастомные инструменты через MCP-серверы и Python-функции.
- **Интеграция с Claude Code:** skill `claude-to-deerflow` позволяет слать задачи в запущенный DeerFlow прямо из терминала Claude Code — установка: `npx skills add https://github.com/bytedance/deer-flow --skill claude-to-deerflow` (github.com/bytedance/deer-flow — claude-to-deerflow SKILL.md).

4. Песочница и файловая система

- Каждая задача получает своё исполнительное окружение с полноценной файловой системой: skills, workspace, uploads, outputs.
- **Docker-режим:** `AioSandboxProvider` запускает shell в изолированных контейнерах — рекомендованный вариант для продакшена.
- **Локальный режим:** `LocalSandboxProvider` мапит файловые операции на директории на хосте, но host-`bash` по умолчанию выключен, так как это не безопасная изоляция.
- **Kubernetes-режим:** выполнение в подах Kubernetes через provisioner-сервис — для тяжёлых многоагентных нагрузок и общего сервера.

5. Память и context engineering

- Долгосрочная память сохраняется между сессиями: профиль, предпочтения, стиль письма, технический стек и повторяющиеся workflow пользователя — хранится локально.
- **Дедупликация:** обновления памяти пропускают повторяющиеся факты, поэтому данные не накапливаются бесконечно от сессии к сессии.
- **Суммаризация:** внутри сессии DeerFlow агрессивно управляет контекстом — суммирует завершённые подзадачи и выгружает промежуточные результаты в файловую систему.
- **Восстановление после сбоя:** при обрыве цепочки tool-call DeerFlow подставляет плейсхолдер-результаты для незавершённых вызовов, чтобы reasoning-модели с жёсткой валидацией истории не падали с ошибкой.

6. Установка и запуск

- **Клонирование:** `git clone https://github.com/bytedance/deer-flow.git`, затем `cd deer-flow` и `make setup` — интерактивный визард выбора LLM-провайдера и режима песочницы занимает около 2 минут.
- **Docker:** `make docker-init` разово подтягивает образ песочницы, `make docker-start` поднимает сервисы для разработки; для продакшена — `make up` и `make down`.
- **Доступ:** после запуска интерфейс открывается по адресу `http://localhost:2026`.
- **Требования для локальной разработки:** Node.js 22+, pnpm, uv и nginx — проверяются командой `make check`.
- **Ресурсы:** для одного разработчика достаточно 4 vCPU / 8 ГБ RAM; для сервера с несколькими агентами и тяжёлыми задачами — от 16 vCPU / 32 ГБ RAM.

7. Интеграции с мессенджерами

- DeerFlow принимает задачи напрямую из чатов — каналы стартуют автоматически при конфигурации, публичный IP не требуется.
- **Поддерживаемые каналы:** Telegram (Bot API), Slack (Socket Mode), Feishu/Lark (WebSocket), WeChat, WeCom, DingTalk.
- **Команды в чате:** `/new` — новый диалог, `/status` — инфо о треде, `/models` — список моделей, `/memory` — просмотр памяти, `/help` — справка.
- **Наблюдаемость:** встроена трассировка через LangSmith и Langfuse — вызовы LLM, агентов и инструментов видны в дашборде.