Explain OpenClaw — Полное руководство

Перевод репозитория centminmod/explain-openclaw

Февраль 2026

Оглавление

Что такое OpenClaw? (простым языком, для начинающих)

Оглавление (Explain OpenClaw)

OpenClaw — это фреймворк для запуска **персонального ИИ-ассистента** на оборудовании под вашим контролем.

Если вы использовали ChatGPT/Claude/Gemini только через браузер:

- эти продукты — *приложения*, которые размещены у кого-то другого
- OpenClaw — это *платформа*, которую вы запускаете сами и подключаете к ней выбранную модель/провайдера

Главное преимущество OpenClaw — не «более умная модель». Оно в том, что OpenClaw может:

- работать **там, где вы уже общаетесь** (WhatsApp, Telegram, Discord, iMessage, …)
- оставаться **постоянно включённым** (сервис Gateway)
- хранить **состояние** (сессии, память, политики)
- при необходимости **выполнять действия** (инструменты, узлы устройств)

Официальный обзор (хорошее введение верхнего уровня): https://docs.openclaw.ai

---

Ключевая идея в одном предложении

> **OpenClaw — это самостоятельно размещаемый Gateway, который соединяет чат-приложения с агентом, способным рассуждать и (опционально) действовать.**

Основные компоненты (с аналогиями)

1) Gateway = «домашняя база вашего ассистента»

Gateway — это долгоживущий процесс, который вы запускаете на подконтрольной вам машине.

Он:

- принимает входящие сообщения от чат-платформ (каналы)
- определяет, что должно произойти (маршрутизация + политики)
- выполняет ход агента (обращается к провайдеру модели)
- отправляет ответ обратно в ту же чат-платформу
- хранит локальное состояние (конфигурацию, учётные данные, логи сессий)

Аналогия:

- маршрутизатор + планировщик + движок политик для вашего ассистента

Документация: https://docs.openclaw.ai/gateway

2) Каналы = «телефонные линии» к ассистенту

Канал — это способ подключения OpenClaw к платформе обмена сообщениями.

Примеры:

- WhatsApp (через WhatsApp Web / Baileys)
- Telegram (Bot API)
- Discord
- iMessage (интеграция с macOS)
- плюс плагины для множества других платформ

Каналы приводят «события входящих сообщений» к единому внутреннему формату.

Документация: https://docs.openclaw.ai/channels

3) Сессии = «память разговора (по умолчанию) на диске»

Сессия — это поток диалога с состоянием:

- история чата (стенограммы)
- метаданные (от кого и откуда пришло сообщение)
- опциональные индексы памяти

Сессии хранятся на диске в каталоге состояния (обычно `~/.openclaw/`).

Документация: https://docs.openclaw.ai/concepts/session

4) Агент = «мозг (модель + правила + политика инструментов)»

Агент — это компонент, в котором фактически вызывается ваша ИИ-модель (Claude/GPT и др.).

OpenClaw предоставляет:

- шаблоны системных промптов
- историю/контекст
- обёртки безопасности
- правила доступности инструментов

Документация: https://docs.openclaw.ai/concepts/agent

5) Инструменты = «руки» (мощные, но рискованные)

Инструменты позволяют модели делать больше, чем просто генерировать текст.

В зависимости от того, что вы включите, инструменты могут включать:

- веб-запросы/поиск
- автоматизацию браузера

- cron/автоматизацию
- exec или вызовы узлов/устройств

Инструменты — основной источник рисков на практике.

Документация: https://docs.openclaw.ai/tools

6) Узлы/устройства = «периферия»

Узлы — это устройства (macOS/iOS/Android), которые могут подключаться к Gateway и предоставлять локальные возможности устройства.

Примеры:

- камера
- аудиовход
- canvas/webview
- (на macOS) удалённое выполнение с подтверждением

Документация: https://docs.openclaw.ai/nodes

Для чего OpenClaw хорошо подходит

Персональный ассистент / ассистент для небольшой команды в чате

- «Подведи итог последних 200 сообщений в этой группе.»
- «Составь ответ в моём стиле.»
- «Веди учёт текущих задач.»

Приватный ассистент с вашими собственными политиками

Поскольку вы контролируете Gateway, вы можете определять:

- кто может с ним общаться
- что он может делать
- куда он может обращаться

Постоянно работающий хост для агента

VPS или домашний сервер может поддерживать Gateway в работе, даже если ваш ноутбук уходит в сон.

Чем OpenClaw *не* является

Не является автоматически безопасным для публичных ботов

Если вы откроете входящие личные сообщения/группы для публики *и* включите инструменты, вы фактически создадите дистанционно управляемый движок автоматизации.

OpenClaw поставляется с функциями безопасности (сопряжение, списки разрешённых, аудиты), но вы должны их использовать.

Документация: https://docs.openclaw.ai/gateway/security

Не является «магией приватности»

OpenClaw хранит состояние локально, но выбранный вами провайдер модели всё равно получает промпты, которые вы отправляете для инференса, — если только вы не используете локальную модель.

Безопасная начальная конфигурация для новичков (рекомендуется)

Если вам нужна высокая приватность и низкий уровень риска:

1) Держите Gateway в режиме **только loopback** (только localhost)
2) Используйте **сопряжение** и/или **списки разрешённых**, чтобы только вы могли его активировать
3) Не включайте мощные инструменты, пока не освоитесь
4) Выполните `openclaw security audit --deep` и устраните найденные проблемы

См. также:

- https://docs.openclaw.ai/gateway/security
- https://docs.openclaw.ai/start/pairing

Далее: Глоссарий | Команды CLI

Глоссарий (простым языком)

Оглавление (Explain OpenClaw)

Этот глоссарий намеренно ориентирован на термины, которые встречаются в выводе CLI и документации.

Агент (Agent)

«Мозг», который превращает входящее сообщение и контекст в ответ. На практике это:

- выбранная модель (провайдер + название модели)
- системный промпт и шаблоны
- правила доступности инструментов
- поведение сессии/памяти

Документация: https://docs.openclaw.ai/concepts/agent

Белый список (Allowlist)

Список идентификаторов, которым разрешено обращаться к боту (через личные сообщения) или взаимодействовать с ним в группах.

Замечание по безопасности: белые списки зачастую являются *настоящей* границей безопасности в ботах для мессенджеров.

Документация: https://docs.openclaw.ai/gateway/security

Привязка (gateway.bind)

Определяет, на каком интерфейсе слушает Gateway:

- `loopback` = только localhost (наиболее безопасный вариант по умолчанию)
- `lan` = интерфейсы локальной сети (требуется аутентификация)
- `tailnet` = привязка только к IP-адресу Tailscale, диапазон 100.x.y.z (требуется Tailscale)

Документация: https://docs.openclaw.ai/gateway/remote и https://docs.openclaw.ai/gateway/tailscale

Канал (Channel)

Коннектор для платформы обмена сообщениями: WhatsApp/Telegram/Discord/iMessage и т.д.

Документация: https://docs.openclaw.ai/channels

Панель управления (Control UI / Dashboard)

Веб-интерфейс, обслуживаемый Gateway (на том же порту, что и WebSocket).

Документация: https://docs.openclaw.ai/web/dashboard

Gateway

Долгоживущий процесс, который отвечает за:

- подключения каналов
- маршрутизацию
- сессии
- плагины
- политику выполнения инструментов
- сопряжение узлов/устройств

Документация: https://docs.openclaw.ai/gateway

Сопряжение (Pairing)

Явное подтверждение со стороны владельца.

Используется для:

- сопряжения через личные сообщения (кто может писать боту)
- сопряжения устройств (какие узлы могут подключаться)

Документация: https://docs.openclaw.ai/start/pairing

Плагин / расширение (Plugin / extension)

Дополнительный код, который выполняется **в том же процессе**, что и Gateway, и добавляет каналы/инструменты/функциональность.

Относитесь к этому как к установке кода в вашего ассистента.

Документация: https://docs.openclaw.ai/plugin и https://docs.openclaw.ai/gateway/security

Сессия (Session)

Сохранённый поток переписки (история + метаданные). По умолчанию сессии хранятся на диске в формате JSONL.

Документация: https://docs.openclaw.ai/concepts/session

Инструмент (Tool)

Возможность, которую модель может вызвать (загрузка веб-страниц, поиск, автоматизация браузера, cron, exec, вызовы узлов).

Документация: https://docs.openclaw.ai/tools

Узел / устройство (Node / device)

Сопутствующее устройство, подключённое к Gateway (iOS/Android/macOS/headless-узлы).

Документация: https://docs.openclaw.ai/nodes

Архитектура (техническое описание, привязанное к данному репозиторию)

Оглавление (Explain OpenClaw)

На этой странице описано, как организована кодовая база OpenClaw и как сообщение превращается в ответ.

Источники проверены по:

- `../docs/index.md` (верхнеуровневое описание)

- `../docs/gateway/index.md` (руководство по Gateway)
- `../docs/gateway/security.md`
- `../src/gateway/server.impl.ts` (запуск Gateway-сервера и валидация конфигурации)
- `../src/auto-reply/reply/agent-runner.ts` (выполнение хода агента)

Высокоуровневый поток данных

```
Мессенджеры (WhatsApp/Telegram/Discord/iMessage/...)
        |
        ▼
Gateway (единый постоянно работающий процесс)
  - приём/отправка сообщений по каналам
  - маршрутизация + проверка политик
  - сессии (на диске)
  - ходы агента
  - вызовы инструментов
        |
        ▼
AI-провайдер(ы) (Anthropic/OpenAI/и др.) ИЛИ локальный эндпоинт модели
        |
        ▼
Обратно в канал
```

OpenClaw построен вокруг **Gateway**. Большинство операций (статус, логи, привязка, отправка, запуск агентов) взаимодействуют с Gateway через его WebSocket RPC.

Gateway — плоскость управления

Gateway:

- занимает один порт (по умолчанию `18789`) и мультиплексирует:
  - WebSocket-плоскость управления
  - UI управления / дашборд
  - опциональные HTTP-эндпоинты (`/v1/chat/completions`, `/v1/responses`, `/tools/invoke`)
- владеет соединениями каналов (сессия WhatsApp Web, long-poll Telegram-бота и т. д.)
- владеет локальным состоянием (конфигурация, учётные данные, транскрипты)

Валидация конфигурации как элемент безопасности

В `src/gateway/server.impl.ts` Gateway:

- считывает снимок конфигурации
- по возможности мигрирует устаревшие записи конфигурации
- отказывается запускаться при невалидной схеме

Это сделано намеренно: неизвестные ключи и некорректные значения расцениваются как небезопасные.

Документация: https://docs.openclaw.ai/gateway/configuration

Ключевые модули в репозитории (где искать)

CLI

- Точка входа: `src/entry.ts` — загружает основной модуль CLI.
- Подключение команд: `src/cli/`.
- Документация CLI для Gateway: `docs/cli/gateway.md`.

Gateway

- Запуск Gateway и основная среда выполнения: `src/gateway/` (точка входа — `src/gateway/server.impl.ts`).
- WebSocket-среда выполнения и методы расположены в модулях `src/gateway/server-*`.

Каналы

- Реализации каналов находятся в отдельных папках (например, `src/telegram`, `src/discord`, `src/imessage`, `src/web` и т. д.).
- Общая логика каналов и вспомогательные функции маршрутизации — в `src/channels/`.

Автоответы / ходы агента

- «Конвейер ответов» находится в `src/auto-reply/`.
- `src/auto-reply/reply/agent-runner.ts` — центральный оркестратор выполнения хода: управление обновлениями сессий, сигналами набора текста, правилами вывода результатов инструментов, повторными запросами и т. д.

Схема конфигурации

- Типы экспортируются из `src/config/types.ts` (разделены на множество файлов `types.*.ts`).
- В документации описаны строгая валидация схемы и её использование в UI.

Жизненный цикл сообщения (подробно)

Ниже представлен концептуальный конвейер. Детали зависят от канала.

1) **Приём (Ingress)**

- Адаптер канала получает событие (личное сообщение / сообщение в группе, вложение, упоминание).

2) **Идентификация и авторизация**

- Определение идентичности отправителя.
- Применение политики личных сообщений (привязка / список разрешённых / открытый доступ / отключено).
- Применение групповой политики (фильтрация по упоминаниям, списки разрешённых, фильтрация по командам).

3) **Решение о маршрутизации**

- Определение целевого агента / ключа сессии.
- Во многих конфигурациях личные сообщения схлопываются в «основную» сессию; группы изолированы.

4) **Сборка контекста**

- Загрузка записей транскрипта / хранилища сессий с диска.
- Формирование контекста промпта агента (шаблоны + системный промпт + история + вложения).

5) **Вызов модели (ход агента)**

- Вызов выбранного провайдера/модели (с логикой резервного переключения, если настроена).
- Потоковая передача событий вывода.
- Если модель запрашивает инструменты — их вызов согласно политике выполнения инструментов Gateway.

6) **Доставка ответа**

- Преобразование вывода агента в сообщения, специфичные для канала.
- Применение правил цепочки ответов, разбиения на части и т. д.
- Отправка обратно в канал.

7) **Сохранение**

- Запись обновлений транскрипта и метаданных на диск.

Порты (что и где слушает)

Базовый порт по умолчанию: `18789`.

На этом порту:

- WebSocket-плоскость управления
- Дашборд (HTTP)

В зависимости от конфигурации могут существовать дополнительные производные порты (управление браузером, хост canvas).

Документация:

- https://docs.openclaw.ai/gateway (руководство по сервису)
- https://docs.openclaw.ai/help/faq (приоритет портов)

Где хранится состояние (взгляд оператора)

Наиболее важная директория — каталог состояния:

- по умолчанию: `~/.openclaw/`
- для профиля: `~/.openclaw-<profile>/`

Типичные пути (см. официальную «карту хранения учётных данных»):

- конфигурация: `~/.openclaw/openclaw.json`
- профили авторизации моделей: `~/.openclaw/agents/<agentId>/agent/auth-profiles.js`
- транскрипты: `~/.openclaw/agents/<agentId>/sessions/*.jsonl`

Документация: https://docs.openclaw.ai/gateway/security

Почему «безопасность» — это в основном конфигурация

Большинство проблем безопасности не являются экзотическими. Типичные причины:

- открытые личные сообщения / группы
- включённые мощные инструменты
- открытая поверхность Gateway без аутентификации
- установка недоверенных плагинов

Именно поэтому существует команда `openclaw security audit` и почему валидация конфигурации настолько строгая.

Документация: https://docs.openclaw.ai/gateway/security

Карта репозитория (где искать в коде)

Оглавление (Explain OpenClaw)

Это практическое руководство по навигации для новых контрибьюторов и читателей.

> Совет: не пытайтесь прочитать всё сразу. Начните с Gateway и проследите путь одного сообщения.

Директории верхнего уровня

Директория	Назначение
`src/`	Исходный код на TypeScript (~49 поддиректорий)
`docs/`	Каноническая документация (источник истины)
`extensions/`	Плагины (workspace-пакеты)
`apps/`	Приложения для macOS/iOS/Android
`packages/`	Общие workspace-пакеты
`ui/`	Веб-интерфейс / дашборд
`skills/`	Определения навыков
`scripts/`	Скрипты сборки, релизов и утилиты
`test/`	Интеграционные / E2E-тесты
`Swabble/`	Интеграция со Swabble
`exports/`	Публичные API-экспорты
`vendor/`	Вендорированный сторонний код
`patches/`	Патч-файлы для зависимостей
`git-hooks/`	Скрипты git-хуков
`assets/`	Статические ресурсы (изображения, иконки)

Ключевые точки входа

CLI

- `src/entry.ts` — точка входа CLI (запускает окружение, затем загружает `src/cli/run-main.js`)
- `src/cli/` — определения CLI-команд
- `src/commands/` — реализации команд (186 файлов)

Gateway

- `src/gateway/server.impl.ts` — запуск Gateway, валидация/миграция конфигурации, связывание компонентов в рантайме
- `src/gateway/server-ws-runtime.ts` (и соседние файлы) — WS-сервер и обвязка RPC-обработчика
- `docs/gateway/index.md` — runbook, соответствующий работе в продакшене

Каналы

- `src/channels/` — общая логика каналов (идентификация, allowlist-ы, гейтинг, реестр)
- Папки каналов с полноценными адаптерами: `src/telegram/`, `src/discord/`, `src/slack/`, `src/signal/`, `src/imessage/`, `src/web/`, `src/line/`, `src/whatsapp/`
- Каналы только с конфигурацией (без директории в `src/`): `googlechat`, `msteams`, `feishu`
- `docs/channels/` — документация по каналам (28 файлов, включая pairing, маршрутизацию, группы)

Ходы агента

- `src/auto-reply/` — конвейер ответов
- `src/auto-reply/reply/agent-runner.ts` — основной оркестратор хода агента
- `src/agents/` — фреймворк агентов (316 файлов): инструменты, песочница, профили авторизации, мульти-агентность

Маршрутизация

- `src/routing/` — выделенный модуль маршрутизации: `session-key.ts`, `resolve-route.ts`, `bindings.ts`

Безопасность

- `src/security/` — логика безопасности (аудит, политики, обёртка внешнего контента)
- `docs/gateway/security/index.md` — модель угроз и чек-лист для операторов

Основные поддиректории `src/`

Директория `src/` содержит ~49 поддиректорий. Ключевые (помимо точек входа, описанных выше):

Директория	Файлов (прибл.)	Назначение
`src/agents/`	~466	Фреймворк агентов, инструменты, песочница, профили авторизации
`src/browser/`	~82	Автоматизация браузера (CDP/Puppeteer)
`src/commands/`	~234	Реализации CLI-команд
`src/config/`	~139	Схема конфигурации, типы, валидация, миграции
`src/cron/`	—	Планирование cron-задач
`src/daemon/`	—	Фоновый процесс-демон
`src/hooks/`	—	Система хуков жизненного цикла
`src/infra/`	~195	Инфраструктура: сеть, защита от SSRF, безопасное выполнение, архивирование

Директория	Файлов (прибл.)	Назначение
src/media/	—	Работа с медиа (загрузка, скачивание, конвертация)
src/media-understanding/	—	Распознавание изображений/аудио/видео с помощью ИИ
src/memory/	—	Управление памятью/контекстом, QMD
src/plugins/	—	Рантайм и загрузка плагинов
src/plugin-sdk/	—	SDK для разработчиков расширений
src/providers/	—	Интеграции с LLM-провайдерами (Anthropic, OpenAI, Ollama и др.)
src/routing/	—	Маршрутизация сообщений, вычисление ключа сессии
src/sessions/	—	Управление сессиями и их компактификация
src/tts/	—	Синтез речи (Text-to-Speech)
src/tui/	—	Терминальный интерфейс
src/wizard/	—	Мастер начальной настройки
src/logging/	—	Логирование, редактирование чувствительных данных
src/link-understanding/	—	Предпросмотр URL / понимание ссылок
src/node-host/	—	Node.js-хост для изолированного выполнения
src/pairing/	—	Процесс привязки устройств
src/process/	—	Управление процессами
src/terminal/	—	Интеграция с терминалом
src/types/	—	Общие TypeScript-типы
src/utils/	—	Общие утилиты
src/shared/	—	Общий межмодульный код
src/acp/	—	ACP (Agent Communication Protocol)
src/canvas-host/	—	Хост для canvas/рисования

Директория	Файлов (прибл.)	Назначение
`src/compat/`	—	Слои совместимости
`src/docs/`	—	Вспомогательные модули встроенной документации
`src/macos/`	—	Нативные интеграции для macOS
`src/markdown/`	—	Обработка Markdown
`src/scripts/`	—	Скрипты уровня исходного кода
`src/test-helpers/`	—	Вспомогательные утилиты для тестов
`src/test-utils/`	—	Утилиты для тестирования

Полезные ripgrep-запросы

Из корня репозитория:

- **Найти, где запускается Gateway:**
  - поиск: `startGatewayServer`
  - файл: `src/gateway/server.impl.ts`
- **Найти логику аудита безопасности:**
  - поиск: `security audit`
  - документация: `docs/gateway/security/index.md`, CLI-документация в `docs/cli/security.md`
- **Найти pairing:**
  - поиск: `pairing` в `src/pairing/` и `docs/channels/pairing.md`
- **Найти поведение allowlist конкретного канала:**
  - поиск `allowFrom` или `dmPolicy` в типах конфигурации каналов (`src/config/types.*.ts` — 29 файлов) и рантайме канала
- **Найти перечисление ChatType / DM policy:**
  - поиск: `ChatType` — используется в адаптерах каналов для маршрутизации групп и личных сообщений
- **Найти санитизацию текста:**
  - поиск: `sanitizeUserFacingText` — используется при нормализации ответов и в инструментах агента
- **Найти логику поиска по памяти:**
  - поиск: `memorySearch` — в `src/config/` и `src/memory/`

Как «проследить путь одного сообщения»

1. Выберите канал (например, Telegram).
2. Найдите его монитор/адаптер в `src/telegram/`.
3. Проследите, как он генерирует нормализованное событие и передаёт его в общую логику каналов/маршрутизации.
4. Модуль `src/routing/` определяет маршрут и вычисляет ключ сессии (`session-key.ts`, `resolve-route.ts`).
5. `src/hooks/` может срабатывать на различных стадиях конвейера (до ответа, после ответа и т. д.).
6. Проследите, как Gateway выбирает ключ сессии.
7. Перейдите в `src/auto-reply/reply/agent-runner.ts`, чтобы увидеть, как выполняется ход агента.
8. Для мульти-агентных сценариев `src/agents/` оркестрирует использование инструментов, суб-агентов и взаимодействие с песочницей.
9. Проследите, как ответ разбивается на части и отправляется обратно.

Документация, которую следует держать открытой при чтении кода

- https://docs.openclaw.ai/start/getting-started
- https://docs.openclaw.ai/gateway
- https://docs.openclaw.ai/gateway/security
- https://docs.openclaw.ai/gateway/configuration
- https://docs.openclaw.ai/help/faq

Руководство по развёртыванию: Standalone Mac mini (local-first, высокая приватность)

Примечание: Данное руководство относится к OpenClaw (ранее Moltbot/Clawdbot).

Содержание (Explain OpenClaw)

Цель: запустить OpenClaw на выделенном Mac mini дома с **минимальным сетевым воздействием**.

Если есть такая возможность — это самый безопасный вариант развёртывания по умолчанию: вы контролируете железо, шифрование диска настраивается легко, а удалённый доступ может быть опциональным.

Официальная документация:

- https://docs.openclaw.ai/start/getting-started
- https://docs.openclaw.ai/gateway/security
- https://docs.openclaw.ai/gateway/remote
- https://docs.openclaw.ai/gateway/tailscale

Рекомендуемая конфигурация (сводка)

- `gateway.bind: "loopback"` (только localhost)
- Политика DM: `pairing` или `allowlist`
- Включайте только те каналы, которые реально нужны
- Не открывайте управление браузером удалённо
- Запускайте `openclaw security audit --deep` после настройки и после каждого изменения конфигурации

Пошаговая установка

1) Создание выделенного пользователя (опционально, но рекомендуется)

Если Mac mini используется как «устройство-ассистент», создайте отдельного пользователя macOS (например, `moltbot`) и запускайте сервис от его имени. Это снижает риск случайной утечки данных в домашний каталог основного пользователя.

Создание пользователя через System Settings или через `dscl` для автоматизации:

```
sudo dscl . -create /Users/moltbot
sudo dscl . -create /Users/moltbot UserShell /bin/zsh
sudo dscl . -create /Users/moltbot UniqueID 550
sudo dscl . -create /Users/moltbot PrimaryGroupID 20
sudo dscl . -create /Users/moltbot NFSHomeDirectory /Users/moltbot
sudo mkdir -p /Users/moltbot
sudo chown moltbot:staff /Users/moltbot
```

2) Установка OpenClaw

```
curl -fsSL https://openclaw.ai/install.sh | bash
```

Или:

```
npm install -g openclaw@latest
```

Проверьте версию Node.js (рекомендуется 22.12.0+ для исправлений безопасности):

```
node --version  # Should be v22.12.0 or later
```

3) Онбординг и установка фоновой службы

```
openclaw onboard --install-daemon
```

Обычно создаётся служба для текущего пользователя (launchd) и записывается конфигурация в `~/.openclaw/`.

Для автоматической/headless установки с пользовательским LLM-провайдером (Ollama, LM Studio и т.д.):

```
export CUSTOM_API_KEY="your-api-key-here"
openclaw onboard --non-interactive --install-daemon \
  --custom-base-url "http://localhost:11434/v1" \
  --custom-model-id "llama3" \
  --custom-compatibility openai
```

Все параметры описаны в флагах неинтерактивного онбординга.

4) Проверка работоспособности

```
openclaw gateway status
openclaw status
openclaw health
openclaw security audit --deep
```

Если аудит предлагает исправления:

```
openclaw security audit --fix
```

5) Открытие дашборда (Control UI)

Локально (на том же компьютере):

- http://127.0.0.1:18789/

Если включена аутентификация и токен ещё не в браузере:

```
openclaw dashboard
```

Подключение каналов обмена сообщениями (обзор)

OpenClaw поддерживает множество каналов; два наиболее распространённых:

WhatsApp

- Использует WhatsApp Web / Baileys.
- Процесс входа обычно через QR-код:

```
openclaw channels login
```

Docs: https://docs.openclaw.ai/channels/whatsapp

Telegram

- Использует токен бота, созданный через @BotFather.
- DM-pairing обычно включён по умолчанию; одобрите себя:

```
openclaw pairing list telegram
openclaw pairing approve telegram <CODE>
```

Docs: https://docs.openclaw.ai/channels/telegram

Опционально: удалённый доступ (всё ещё приватный)

Вариант A (универсальный): SSH-туннель

С вашего ноутбука:

```
ssh -N -L 18789:127.0.0.1:18789 user@mac-mini
```

Затем откройте:

- http://127.0.0.1:18789/

Вариант B (лучший UX): Tailscale Serve

Оставьте `gateway.bind: "loopback"` и используйте Tailscale Serve для публикации Control UI в вашей tailnet через HTTPS.

```
brew install tailscale
tailscale up
sudo tailscale serve --bg --https=443 127.0.0.1:18789
```

```
{
  "gateway": {
    "bind": "loopback",
    "tailscale": { "mode": "serve" },
    "auth": { "allowTailscale": true }
  }
}
```

Доступ по адресу `https://<machine-name>.<tailnet>.ts.net/`.

Подробнее о shields-up, ACL и Funnel см. в разделе Tailscale для VPS.

Docs: https://docs.openclaw.ai/gateway/tailscale

После развёртывания: прочитайте это в первую очередь

> **Прежде чем начать ежедневно использовать OpenClaw**, ознакомьтесь с типичными проблемами реальных пользователей:
>
> - **Правило 60% успеха** — задачи из >10 шагов проваливаются в 40% случаев из-за дрифта контекста
> - **Неоднозначность «черновик vs отправка»** — агенты могут интерпретировать «создай черновик» как «создай и отправь»
> - **Утечка профиля браузера** — использование рабочего профиля Chrome даёт агенту доступ ко ВСЕМ вашим залогиненным аккаунтам
> - **Ловушка бездействия** — длительные сессии приводят к зависанию агента или потере контекста
> - **Стоимость Always-On** — круглосуточная работа стоит дороже ожидаемого (473 запроса/день = $847/мес в одном случае)
>
> См.: Операционные подводные камни — 10 реальных паттернов использования, которые идут не так, и способы их исправления.

Чек-лист защиты хоста (Mac mini)

На основе VibeProof Security Guide (использует устаревшее название «Moltbot») и ревью кода.

Операционная система

- ☐ Включить FileVault (полное шифрование диска)
- ☐ Обновлять macOS: `softwareupdate -ia`
- ☐ Включить файрвол: System Settings ☒ Network ☒ Firewall ☒ Turn On

Изоляция пользователя

- ☐ Создать выделенного пользователя (см. шаг 1 выше)
- ☐ Запускать службу Gateway от имени этого пользователя

Версия Node.js

Убедитесь, что установлена Node.js 22.12.0+ (включает критические исправления безопасности):

```
node --version  # Should be v22.12.0 or later
```

Безопасность Gateway

Установите токен аутентификации Gateway для продакшена:

```
export GATEWAY_AUTH_TOKEN="$(openssl rand -hex 32)"
# Add to ~/.zprofile or pass via config
```

Защита учётных данных

- ☐ Относиться к `~/.openclaw/` как к секретному материалу (права 0700)
- ☐ Не устанавливать случайные глобальные npm-пакеты

Защита истории shell от утечки учётных данных:

```
# Add to ~/.zshrc or ~/.zprofile
export HISTCONTROL=ignoreboth
export HISTFILESIZE=0
```

Настройка песочницы

Включите Docker-песочницу для инструментов выполнения кода:

```
# In openclaw.json
# "agents.defaults.sandbox": "docker"
# "agents.defaults.sandboxNetwork": "none"
```

Это изолирует любую успешную prompt-инъекцию в контейнерной среде.

> **См.:** Атаки с помощью prompt-инъекций — 27 примеров паттернов атак, от которых защищает изоляция в песочнице.

Резервные копии (приватность прежде всего)

Если делаете бэкапы — копируйте только то, что понимаете, и шифруйте.

Рекомендуется копировать:

- конфигурацию `openclaw.json`
- только те учётные данные, которые вы готовы восстанавливать

Не рекомендуется копировать:

- транскрипты сессий (если только они не нужны явно)

Docs: https://docs.openclaw.ai/gateway/security

Руководство по развёртыванию: Изолированный VPS (удалённый + защищённый)

> **Примечание:** Это руководство предназначено для OpenClaw (ранее Moltbot/Clawdbot).

Обозначения уровней навыков

Тег	Значение
`[All]`	Это должен сделать каждый
`[Intermediate]`	Рекомендуется для пользователей, уверенно работающих с CLI Linux
`[Advanced]`	Дополнительное усиление безопасности для развёртываний с акцентом на защиту

Оглавление (Explain OpenClaw)

        - DigitalOcean 1-Click Deploy
        - Настройка Tailscale
        - Усиление обратного прокси
        - Терминация TLS
        - Зашифрованное хранилище секретов
        - Усиление Docker-развёртывания
        - Усиление конфигурации OpenClaw
        - Ротация токена шлюза
      - Cloudflare Moltworker
      - Docker Model Runner
- Оптимизации
  - Стоимость + оптимизация токенов
- Справочник
  - Команды + устранение неполадок

Цель: запустить шлюз (Gateway) на Linux VPS, сохраняя **приватный** доступ и усиленную защиту хоста.

Эта конфигурация подходит, когда:

- вы хотите, чтобы ассистент работал постоянно
- ваш ноутбук часто уходит в спящий режим
- вам нужна предсказуемая сетевая конфигурация

Однако требования к безопасности выше: VPS — это машина, подключённая к интернету.

> **Рекомендуемый путь:** Для большинства пользователей следует выполнить разделы 1–6 (базовая настройка), затем перейти к разделу 12 (Tailscale). Tailscale обеспечивает зашифрованный доступ, автоматический TLS и нулевое количество открытых портов — без необходимости настраивать nginx, certbot или TLS вручную. Разделы 13–18 предназначены для продвинутых пользователей, которым нужен публичный доступ или дополнительное усиление.

Связанная официальная документация:

- https://docs.openclaw.ai/gateway/remote
- https://docs.openclaw.ai/gateway/security
- https://docs.openclaw.ai/platforms/linux
- https://docs.openclaw.ai/help/faq

После развёртывания: прочитайте в первую очередь

Прежде чем начать ежедневно использовать OpenClaw, ознакомьтесь с этими практическими проблемами от реальных пользователей:

- **Правило 60% успеха** — задачи с более чем 10 шагами проваливаются в 40% случаев из-за дрейфа контекста
- **Неоднозначность «черновик vs отправить»** — агенты могут интерпретировать «создай черновик» как «создай и отправь»
- **Утечка профиля браузера** — использование вашего повседневного профиля Chrome даёт агенту доступ ко ВСЕМ вашим авторизованным аккаунтам
- **Ловушка бездействия** — длительные сессии приводят к тому, что агент зависает или теряет контекст
- **Стоимость постоянной работы** — круглосуточная работа обходится дороже, чем ожидалось (473 запроса/день = $847/месяц в одном случае)
- **Щит Nginx** — ограничение частоты запросов КРИТИЧЕСКИ важно для предотвращения неконтролируемых расходов на API

Подробнее: Практические проблемы эксплуатации — 10 реальных паттернов использования, которые идут не так, и способы их устранения.

Рекомендуемая позиция безопасности (сводка)

- Шлюз должен слушать **только на loopback** (`gateway.bind: "loopback"`).
- Доступ через **Tailscale** (рекомендуется) или SSH-туннель (запасной вариант).
- Использовать аутентификацию по токену/паролю — или аутентификацию на основе идентификации Tailscale.
- Запускать под выделенным пользователем без прав root.
- Ограничить права доступа к файлам.
- Отключить mDNS-обнаружение на VPS.

---

1) Подготовка VPS (базовое усиление) `[All]`

На основе VibeProof Security Guide (используется устаревшее название «Moltbot»).

Изоляция провайдера: Используйте **отдельного VPS-провайдера** (или как минимум отдельный аккаунт) от того, который вы уже используете для продакшн-сервисов. Если OpenClaw нарушит условия обслуживания — исходящий спам через инъекцию промптов, неправильно настроенный шлюз, передающий вредоносный трафик, неконтролируемые вызовы API или нарушения модерации контента от сообщений, сгенерированных ИИ, — провайдер может **заблокировать весь ваш аккаунт**, а не только VPS с OpenClaw.

Это означает, что все ваши другие инстансы, снапшоты, резервные копии и DNS-записи в этом аккаунте тоже станут недоступны. Одноразовый Droplet за $6/месяц не стоит того, чтобы рисковать продакшн-инфраструктурой.

См. также: Риски VPS — Радиус поражения на уровне аккаунта для полной модели угроз.

Выбор провайдера

- AWS EC2: t3.small, Ubuntu 24.04 LTS
- DigitalOcean: Basic $6/month Droplet, Ubuntu 24.04
- Linode: Nanode 1GB, Ubuntu 24.04
- Hetzner: CX11, Ubuntu 24.04

Настройка SSH-ключа (если у вас его нет)

Если у вас ещё нет пары SSH-ключей, сгенерируйте её на вашей **локальной машине** (не на сервере):

```
# Generate an Ed25519 key (recommended — stronger than RSA, shorter keys)
ssh-keygen -t ed25519 -C "your-email@example.com"

# Or RSA 4096-bit if Ed25519 is not supported
ssh-keygen -t rsa -b 4096 -C "your-email@example.com"
```

При запросе парольной фразы **установите её** — она защитит ваш ключ в случае компрометации ноутбука.

Большинство облачных провайдеров позволяют загрузить ваш публичный ключ (`~/.ssh/id_ed25519.pub`) при создании VPS. Если ваш провайдер этого не поддерживает, скопируйте его после первого входа:

```
# Copies your public key to the server so you can log in without a password
ssh-copy-id -i ~/.ssh/id_ed25519.pub ubuntu@YOUR_SERVER_IP
```

Первоначальная настройка

```
# Log in to your VPS using your SSH key
ssh -i ~/.ssh/id_ed25519 ubuntu@YOUR_SERVER_IP

# Download the latest package lists, then install all available updates
sudo apt update && sudo apt upgrade -y

# Create a dedicated user account for running OpenClaw (not root)
sudo adduser openclaw
# Give that user permission to run admin commands with "sudo"
sudo usermod -aG sudo openclaw
```

Настройка межсетевого экрана (критически важно)

Разрешите SSH только с вашего IP. **Никогда не разрешайте доступ с `0.0.0.0/0` (ото-всюду).**

```
# Block ALL incoming connections by default (nothing gets in unless you
↪  allow it)
sudo ufw default deny incoming
# Allow all outgoing connections (your server can still reach the internet)
sudo ufw default allow outgoing
# Punch a hole for SSH (port 22) so you don't lock yourself out
sudo ufw allow ssh
# Turn the firewall on — defaults must be set FIRST
sudo ufw enable

# Show the current firewall rules to confirm everything looks right
sudo ufw status
```

Облачный межсетевой экран: Также настройте межсетевой экран вашего провайдера (Security Groups на AWS и т.д.), чтобы разрешить SSH только с вашего IP. **Не** открывайте порт 18789 для публичного интернета.

Усиление SSH (критически важно)

Отключите аутентификацию по паролю и вход под root для предотвращения атак перебором:

```
# Open the SSH server config file in a text editor
sudo nano /etc/ssh/sshd_config
```

Найдите и установите (или раскомментируйте) эти строки:

```
PermitRootLogin no              # Nobody can SSH in as "root" directly
PasswordAuthentication no       # Only key-based login allowed (no password gues
PubkeyAuthentication yes        # Enable SSH key authentication
```

Перед перезагрузкой проверьте конфигурацию, чтобы не потерять доступ:

```
# Dry-run the SSH config — catches typos before they take effect
# If you see no output, the config is valid
sudo sshd -t
```

Если тест прошёл без вывода, перезагрузите SSH для применения изменений:

```
# Tell the SSH service to re-read its config (existing connections stay
↪  open)
sudo systemctl reload ssh
```

Предотвращение блокировки: Откройте **второй терминал** и проверьте SSH-вход, прежде чем закрывать текущую сессию. Если новое подключение работает — всё в порядке. Если нет — исправьте `sshd_config` в ещё открытой сессии.

> **Совет для начинающих:** Если вы всё-таки потеряли доступ, используйте веб-консоль вашего провайдера (вкладка «Access» в DigitalOcean, «EC2 Instance Connect» в AWS и т.д.) для исправления конфигурации.

Настройка файла подкачки

Небольшие VPS-инстансы (1–2 ГБ ОЗУ) могут исчерпать память во время сессий агента, что приводит к завершению процесса (OOM). Файл подкачки служит аварийным буфером:

```
# Allocate a 2GB file on disk to use as swap (virtual memory)
sudo fallocate -l 2G /swapfile
# Only root should read/write the swap file (security best practice)
sudo chmod 600 /swapfile
# Format the file as swap space
sudo mkswap /swapfile
# Activate the swap immediately
sudo swapon /swapfile

# Add it to /etc/fstab so it activates automatically after a reboot
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab

# Confirm swap is active — you should see "Swap: 2.0G" in the output
free -h
```

Синхронизация времени NTP `[Intermediate]`

Точные временные метки необходимы для корреляции логов безопасности и отладки. Большинство облачных провайдеров синхронизируют время автоматически, но проверьте и установите chrony при необходимости:

```
# Check if your system clock is being synced — look for "NTP service:
↪  active"
timedatectl status

# If NTP is inactive, install chrony (a lightweight time sync daemon)
sudo apt install chrony
# Start chrony on boot
sudo systemctl enable chrony

# Confirm it's syncing — "Leap status: Normal" means it's working
chronyc tracking
```

---

2) Установка OpenClaw `[All]`

На VPS:

```
# Download and run the official OpenClaw installer
# (installs the openclaw binary and its dependencies including Node.js)
curl -fsSL https://openclaw.ai/install.sh | bash
```

Проверьте версию Node.js (рекомендуется 22.12.0+ для получения патчей безопасности):

```
node --version  # Should be v22.12.0 or later
```

Затем выполните начальную настройку — это пошаговый мастер, который также создаёт фоновый сервис:

```
# Interactive setup wizard + installs a systemd service so the gateway
# starts automatically on boot
openclaw onboard --install-daemon
```

Установите токен аутентификации шлюза для продакшна:

```
# Generate a random 64-character hex token (cryptographically secure)
export GATEWAY_AUTH_TOKEN="$(openssl rand -hex 32)"
# Save it to your shell profile so it persists across logins
echo "export GATEWAY_AUTH_TOKEN='$GATEWAY_AUTH_TOKEN'" >> ~/.profile
```

Если вы работаете без графического интерфейса и вам нужна OAuth-аутентификация, выполните шаг авторизации на доверенной машине и скопируйте необходимые файлы учётных данных, как описано в документации.

Документация: https://docs.openclaw.ai/start/getting-started

Неинтерактивная начальная настройка (CI/CD и автоматизация)

Для автоматизированных развёртываний или пользовательских LLM-провайдеров (Ollama, LM Studio, LiteLLM proxy и т.д.) используйте неинтерактивный режим:

```
# Using env var for API key (recommended — avoids process list exposure)
export CUSTOM_API_KEY="your-api-key-here"
openclaw onboard --non-interactive --install-daemon \
  --custom-base-url "https://llm.example.com/v1" \
  --custom-model-id "my-model" \
  --custom-compatibility openai

# Or with explicit auth choice and all options
openclaw onboard --non-interactive --install-daemon \
  --auth-choice custom-api-key \
  --custom-base-url "http://localhost:11434/v1" \
  --custom-model-id "llama3" \
  --custom-provider-id "ollama" \
  --custom-compatibility openai
```

Доступные флаги `--custom-*`:

Флаг	Обязательный	Описание
`--custom-base-url <url>`	Да	URL эндпоинта провайдера
`--custom-model-id <id>`	Да	Идентификатор модели
`--custom-api-key <key>`	Нет	API-ключ (резервный вариант: переменная окружения `CUSTOM_API_KEY`, затем профиль аутентификации)

Флаг	Обязательный	Описание
`--custom-provider-id <id>`	Нет	Идентификатор провайдера (автоматически определяется из URL, если не указан)
`--custom-compatibility <mode>`	Нет	`openai` (по умолчанию) или `anthropic`

> **Безопасность:** Избегайте передачи API-ключей через флаг `--custom-api-key` в общих средах — они видны в списке процессов. Вместо этого используйте переменную окружения `CUSTOM_API_KEY`.

Документация: https://docs.openclaw.ai/start/wizard-cli-automation

---

3) Оставьте шлюз только на loopback `[All]`

Это самый безопасный паттерн удалённого доступа:

- Шлюз слушает только на `127.0.0.1:18789`.
- Вы пробрасываете порт, когда нужен доступ.

---

4) Удалённый доступ (рекомендуемый) `[All]`

Вариант A: Tailscale (рекомендуется)

Tailscale — рекомендуемый способ доступа к вашему VPS. Он обеспечивает зашифрованную передачу данных, аутентификацию на основе идентификации и автоматический TLS — с нулевым количеством портов, открытых в публичный интернет. nginx и certbot не требуются.

Установите Tailscale и на VPS, и на ваши персональные устройства, затем используйте Tailscale Serve для публикации шлюза по HTTPS в вашей приватной сети tailnet. Полное пошаговое описание в разделе 12.

Документация: https://docs.openclaw.ai/gateway/tailscale

Вариант B: SSH-туннель (универсальный запасной вариант)

Если вы не можете использовать Tailscale (например, из-за корпоративной политики, отсутствия аккаунта Tailscale), SSH-туннелирование работает везде:

```
# Create an SSH tunnel: forwards your laptop's port 18789 to the VPS's port
↪  18789
# -N means "don't open a shell, just forward the port"
# This must stay running — closing the terminal closes the tunnel
ssh -N -L 18789:127.0.0.1:18789 user@gateway-host
```

Теперь ваш локальный браузер может открыть:

- http://127.0.0.1:18789/

...и ваш локальный CLI может взаимодействовать со шлюзом по проброшенному URL. Недостаток в том, что SSH-сессия должна оставаться открытой, и нет автоматического TLS или аутентификации на основе идентификации.

Документация: https://docs.openclaw.ai/gateway/remote

---

5) Проверка и блокировка [All]

На VPS:

```
# Check if the gateway process is running and listening
openclaw gateway status
# Show overall OpenClaw status (all components)
openclaw status
# Run a health check — confirms the service is responding
openclaw health
# Deep security scan — checks config, permissions, and known issues
openclaw security audit --deep
```

Если аудит обнаружит проблемы, многие из них можно исправить автоматически:

```
# Automatically apply recommended security fixes
openclaw security audit --fix
```

---

6) Логирование + устранение неполадок на сервере [All]

Типичный процесс устранения неполадок:

```
# Show status of all OpenClaw components (gateway, agents, channels)
openclaw status --all
# Stream live logs — useful for watching what happens in real time (Ctrl+C
↪  to stop)
openclaw logs --follow
```

Если сервис выглядит запущенным, но проверка завершается ошибкой:

- возможно, несоответствие профиля/конфигурации
- или процесс жив, но не слушает порт

Документация: https://docs.openclaw.ai/help/faq

---

7) Рекомендации по безопасности VPS [All]

- Минимизируйте количество плагинов.
- Используйте отдельные учётные записи мессенджеров для бота.
- Относитесь к эндпоинтам управления браузером как к административным API.
- Ротируйте токены и API-ключи при подозрении на утечку.
- Ограничьте права доступа к `~/.openclaw/` (`chmod 700 ~/.openclaw`).

Отключение mDNS/Bonjour-обнаружения

По умолчанию шлюз OpenClaw анонсирует своё присутствие в локальной сети с помощью mDNS (Multicast DNS, также называемого Bonjour). Он рекламирует сервис `_openclaw-gw._tcp`, чтобы клиенты OpenClaw (десктопное приложение, мобильные приложения) могли автоматически обнаруживать шлюз без ручной настройки.

На VPS это представляет проблему:

- **Виртуальный хостинг**: другие арендаторы в том же сетевом сегменте могут увидеть, что OpenClaw запущен, и узнать ваше имя хоста.
- **Выделенный VPS**: mDNS работает только в пределах локального канала (не проходит через маршрутизаторы), поэтому риск ниже, но пользы тоже нет — вы подключаетесь через SSH-туннель или Tailscale, а не через локальное обнаружение.

Отключите, добавив в ваш `~/.profile` или `~/.bashrc`:

```
export OPENCLAW_DISABLE_BONJOUR=1
```

Или установите в `openclaw.json`:

```
{
  "discovery": {
    "mdns": { "mode": "off" }
  }
}
```

Источник: `src/infra/bonjour.ts:29`, `src/gateway/server-discovery-runtime.ts:27`.
Перекрёстные ссылки: Модель угроз — mDNS/Bonjour, Чек-лист усиления, раздел 10.

Защита истории команд от утечки учётных данных

```
# Add to ~/.profile or ~/.bashrc
# ignoreboth = don't save commands that start with a space OR duplicate
 ↪  commands
export HISTCONTROL=ignoreboth
# Set history file size to 0 = don't write any commands to the on-disk
 ↪  history file
# (prevents tokens/keys you typed from being saved to ~/.bash_history)
export HISTFILESIZE=0
```

Документация: https://docs.openclaw.ai/gateway/security

---

8) Автоматические обновления безопасности [All]

Поддерживайте систему автоматически обновлённой:

```
# Install the automatic update tool
sudo apt install unattended-upgrades
# Interactive setup — select "Yes" to enable automatic security updates
sudo dpkg-reconfigure unattended-upgrades
```

Это гарантирует автоматическое применение критических патчей безопасности без ручного вмешательства.

---

9) Усиление SSH с помощью Fail2ban [All]

Защита от атак перебором на SSH:

```
# Install fail2ban — monitors SSH login attempts and bans IPs after too
↪  many failures
sudo apt install fail2ban
# Start fail2ban on boot
sudo systemctl enable fail2ban
# Start it now
sudo systemctl start fail2ban
```

Проверка статуса:

```
# Shows how many IPs are currently banned and total failed attempts
sudo fail2ban-client status sshd
```

---

10) Systemd-сервис и ограничения ресурсов [Intermediate]

Команда `openclaw onboard --install-daemon` создаёт пользовательский systemd-сервис. Если вам нужно настроить его или создать вручную, вот полный файл сервиса.

Полный файл пользовательского systemd-сервиса

Создайте `~/.config/systemd/user/openclaw-gateway.service`:

```
[Unit]
Description=OpenClaw Gateway
After=network-online.target
Wants=network-online.target

[Service]
```

```
Type=simple
ExecStart=%h/.openclaw/bin/openclaw gateway --foreground
Restart=on-failure
RestartSec=5
Environment=NODE_ENV=production

# Resource limits (adjust for your VPS size)
MemoryMax=1G
CPUQuota=80%

# Security hardening
NoNewPrivileges=true
ProtectSystem=strict
ProtectHome=read-only
ReadWritePaths=%h/.openclaw

[Install]
WantedBy=default.target
```

Активация и запуск:

```
# Tell systemd to re-read service files (picks up your new/changed file)
systemctl --user daemon-reload
# Start the service automatically on boot
systemctl --user enable openclaw-gateway
# Start it right now
systemctl --user start openclaw-gateway

# By default, user services stop when you log out. "Linger" keeps them
# running even when you're not SSH'd in — essential for an always-on server
loginctl enable-linger $(whoami)
```

Системный сервис (альтернатива)

Если вы запускаете OpenClaw как системный сервис (например, `/etc/systemd/system/openclaw-ga
добавьте те же ограничения ресурсов:

```
[Service]
MemoryMax=1G
CPUQuota=80%
```

Затем перезагрузите:

```
# Re-read service files to pick up changes
sudo systemctl daemon-reload
# Restart the service with new limits applied
sudo systemctl restart openclaw-gateway
```

Это предотвращает потребление всех системных ресурсов неконтролируемыми процессами.

10a) Резервное копирование и восстановление [All]

OpenClaw хранит конфигурацию, учётные данные и данные сессий в `~/.openclaw/`. Регулярно создавайте резервные копии.

Создание резервной копии

```
# Full backup — compresses the entire .openclaw/ directory into a dated
 ↪  archive
# "tar czf" = create (c) a gzip-compressed (z) archive file (f)
tar czf openclaw-backup-$(date +%Y%m%d).tar.gz -C ~/ .openclaw/

# Privacy-conscious backup — same thing but skips chat logs and temp files
tar czf openclaw-backup-$(date +%Y%m%d).tar.gz \
  --exclude='.openclaw/sessions' \
  --exclude='.openclaw/workspace' \
  -C ~/ .openclaw/
```

Восстановление из резервной копии

```
# Stop the gateway so files aren't being written while we restore
systemctl --user stop openclaw-gateway

# Extract the backup archive into your home directory
# "tar xzf" = extract (x) a gzip-compressed (z) archive file (f)
tar xzf openclaw-backup-YYYYMMDD.tar.gz -C ~/

# Lock down permissions — 700 means only your user can access the directory
chmod 700 ~/.openclaw
# 600 means only your user can read/write the config file (contains
 ↪  secrets)
chmod 600 ~/.openclaw/openclaw.json

# Start the gateway again with the restored config
systemctl --user start openclaw-gateway
```

Рекомендации

- Храните резервные копии в **зашифрованном месте** (например, том LUKS, зашифрованный S3 bucket). Резервная копия содержит API-ключи и токены в открытом виде.
- Запланируйте еженедельное резервное копирование через cron:

```
# Add a weekly cron job: runs every Sunday at 2 AM, creates a dated backup
 ↪  archive
echo "0 2 * * 0 $(whoami) tar czf
 ↪  /home/$(whoami)/backups/openclaw-backup-\$(date +\%Y\%m\%d).tar.gz -C
 ↪  /home/$(whoami)/ .openclaw/" | sudo tee -a /etc/crontab > /dev/null
```

10b) Браузерный агент на безголовом VPS [Intermediate]

Если вы хотите, чтобы браузерный агент OpenClaw работал на безголовом VPS, вам нужна установка headless Chromium. OpenClaw управляет процессом браузера самостоятельно — **не** запускайте отдельный сервис Chrome/CDP.

Установка headless Chromium

```
# Install Chromium from Ubuntu's package manager (gets automatic security
↪  updates)
sudo apt install chromium-browser

# Confirm it installed — prints the version number
chromium-browser --version
```

Настройка OpenClaw

Добавьте в `openclaw.json`:

```
{
  "browser": {
    "evaluateEnabled": false
  }
}
```

`evaluateEnabled: false` предотвращает произвольное выполнение JavaScript в контексте браузера — полное усиление конфигурации описано в разделе 17.

Чего НЕ следует делать

- **Не** устанавливайте Chrome через Homebrew, snap или отдельные `.deb`-пакеты — `apt` проще и автоматически получает обновления безопасности.
- **Не** запускайте отдельный сервис Chrome DevTools Protocol (CDP) на фиксированном порту (например, `--remote-debugging-port=18800`). Это открывает высокопривилегированный отладочный интерфейс. OpenClaw запускает и управляет своим собственным экземпляром браузера.
- **Не** устанавливайте сторонние пакеты управления браузером (`agent-browser` и т.д.) — шлюз OpenClaw управляет жизненным циклом браузера самостоятельно.

---

11) Развёртывание DigitalOcean 1-Click [All]

> **Самый быстрый путь к защищённому VPS.** Приложение DigitalOcean Marketplace автоматически настраивает лучшие практики безопасности.

Официальные ресурсы:

- Marketplace: https://marketplace.digitalocean.com/apps/moltbot
- Руководство: https://www.digitalocean.com/community/tutorials/how-to-run-moltbot

Что 1-Click настраивает автоматически

Развёртывание 1-Click настраивает следующие меры безопасности из коробки:

Мера	Описание
Аутентифицированный токен шлюза	Автоматически сгенерирован; защищает от несанкционированного доступа
Усиленные правила межсетевого экрана	Ограничение частоты запросов на портах OpenClaw для предотвращения DoS
Запуск без прав root	OpenClaw работает под непривилегированным пользователем, уменьшая поверхность атаки
Изоляция в Docker-контейнере	Изолированная среда выполнения
Приватное сопряжение через личные сообщения	Включено по умолчанию; предотвращает несанкционированный обмен сообщениями

Системные требования

Уровень использования	ОЗУ	CPU	Рекомендуется для
Персональный (1–5 пользователей)	4GB	2	Индивидуальное использование, мало каналов
Малая команда (5–20)	8GB	4	Несколько каналов
Средняя команда (20–50)	16GB	8	Интенсивное использование
Большая команда (50+)	32GB	16	Развёртывание с высокой нагрузкой

Шаг 1: Создание Droplet

Через консоль DigitalOcean:

1. Войдите в DigitalOcean -> **Create Droplet**
2. В разделе **Choose an Image** -> вкладка **Marketplace**
3. Найдите «OpenClaw» и выберите его
4. Выберите как минимум **4GB RAM** (s-2vcpu-4gb или выше)
5. Добавьте ваш SSH-ключ в разделе **Authentication**
6. Задайте имя хоста (например, `openclaw-server`)
7. Нажмите **Create Droplet**

Через API:

```
curl -X POST -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer '$TOKEN'' -d \
  ↪  '{"name":"openclaw-server","region":"nyc3","size":"s-2vcpu-4gb","image":"moltbot"}
  ↪  \
  "https://api.digitalocean.com/v2/droplets"
```

Шаг 2: Подключение и настройка

Дождитесь завершения подготовки Droplet (панель может показать «ready» до того, как SSH станет доступен — повторите попытку через 60 секунд при необходимости).

```
ssh root@your_droplet_ip
```

Приветственное сообщение отображает **URL панели управления** — сохраните его для доступа через браузер.

Интерактивная настройка:

1. Выберите LLM-провайдера: Anthropic (рекомендуется), Gradient или OpenAI (скоро)
2. Вставьте ваш API-ключ по запросу
3. Сервис clawdbot перезапускается автоматически

Шаг 3: Доступ к OpenClaw

Терминальный интерфейс (TUI):

```
/opt/clawdbot-tui.sh
```

Веб-панель управления:

Откройте URL панели управления из приветственного сообщения в вашем браузере. URL включает токен шлюза для аутентификации.

Шаг 4: Добавление каналов обмена сообщениями

WhatsApp:

```
/opt/clawdbot-cli.sh channels add
# Select WhatsApp -> scan the QR code with your phone
```

Telegram:

1. Запустите `/opt/clawdbot-cli.sh channels add` и выберите Telegram
2. Откройте Telegram -> найдите @BotFather -> отправьте `/newbot`
3. Следуйте инструкциям для создания бота и получения токена
4. Вставьте токен бота обратно в CLI
5. Откройте URL панели управления и добавьте ваш Telegram user ID в список разрешённых
6. Начните общение с вашим ботом

Ручная настройка

Отредактируйте `/opt/clawdbot.env` для настройки провайдера, шлюза и каналов:

```
nano /opt/clawdbot.env
systemctl restart clawdbot
```

Команды для устранения неполадок

Задача	Команда
Проверить статус сервиса	`systemctl status clawdbot`
Просмотр логов в реальном времени	`journalctl -u clawdbot -f`
Редактирование переменных окружения	`nano /opt/clawdbot.env`
Запуск TUI	`/opt/clawdbot-tui.sh`

Отличия от ручной настройки VPS

1-Click автоматически выполняет разделы 1–3 этого руководства:

- Подготовка VPS с Ubuntu 24.04 + Node.js 22 + Docker
- Установка OpenClaw и настройка сервиса
- Настройка шлюза с токеном аутентификации

Вам всё ещё нужно:

- Настроить каналы обмена сообщениями (Шаг 4 выше)
- При необходимости добавить SSH-туннель или Tailscale для удалённого доступа (см. раздел 4)
- Просмотреть Чек-лист безопасности ниже

Ресурсы

- **Документация OpenClaw:** https://docs.openclaw.ai/
- **Настройка шлюза:** https://docs.openclaw.ai/gateway/configuration
- **Настройка каналов:** https://docs.openclaw.ai/channels
- **Руководство по безопасности:** https://docs.openclaw.ai/gateway/security
- **Сообщество Discord:** https://discord.gg/molt
- **GitHub:** https://github.com/openclaw/openclaw

---

12) Настройка Tailscale (рекомендуется) `[Intermediate]`

> Если вы используете Tailscale, вы можете полностью пропустить разделы 13 (обратный прокси) и 14 (TLS) — Tailscale автоматически обеспечивает зашифрованную передачу данных и аутентификацию на основе идентификации.

Для большинства однопользовательских VPS-развёртываний Tailscale проще и безопаснее, чем путь с обратным прокси nginx + certbot. Он обеспечивает встроенный TLS, аутентификацию на основе идентификации и нулевое количество открытых портов.

12.1) На стороне VPS: Установка и аутентификация [All]

```
# Download and install the Tailscale VPN client
curl -fsSL https://tailscale.com/install.sh | sh

# Connect to your Tailscale account — prints a URL you open in a browser to
 ↪  log in
sudo tailscale up
```

Для **безголового VPS** без браузера сгенерируйте ключ аутентификации на странице Keys в консоли администрирования Tailscale, затем:

```
# Log in without a browser using a pre-generated auth key (replace XXXXX
 ↪  with your key)
sudo tailscale up --auth-key=tskey-auth-XXXXX
```

Проверьте подключение:

```
# Shows all devices on your tailnet and their connection status
tailscale status
# Prints just this machine's tailnet IP address (starts with 100.x.y.z)
tailscale ip -4
```

Опционально: Tailscale SSH — Чтобы полностью убрать порт 22 (а не просто закрыть его межсетевым экраном), включите Tailscale SSH:

```
# Enables SSH access over Tailscale using identity-based auth (no SSH keys
 ↪  needed)
sudo tailscale up --ssh
```

Это позволяет пользователям tailnet подключаться к VPS по SSH без прослушивания `sshd` на публичном порту. Затем вы можете убрать правила UFW для порта 22 или полностью отключить `sshd`. Tailscale SSH отделён от Tailscale Serve (который обслуживает UI шлюза) — оба можно использовать одновременно.

Проверьте с вашей локальной машины: `ssh user@100.x.y.z` должен работать без вашего SSH-ключа.

12.2) На стороне клиента: Установка на персональные устройства [All]

Установите Tailscale на каждое устройство, с которого вы будете подключаться к VPS:

Платформа	Способ установки
macOS	Скачайте с https://tailscale.com/download или `brew install tailscale`
Windows	Скачайте с https://tailscale.com/download
Linux	`curl -fsSL https://tailscale.com/install.sh | sh`
iOS/Android	App Store / Play Store

После установки на каждом устройстве:

1. Откройте Tailscale и войдите с тем **же провайдером идентификации** (Google, Microsoft, Apple), который вы использовали для VPS
2. Убедитесь, что устройство появилось на странице Machines в консоли администрирования
3. Проверьте доступность VPS по его tailnet IP: `ping 100.x.y.z`

12.3) Shields-up на персональных устройствах `[All]`

`shields-up` блокирует **все** входящие Tailscale-подключения к устройству. Включите на ноутбуках и телефонах, которым нужно только *инициировать* подключения к VPS, но не принимать их:

```
# Run this on your LAPTOP/PHONE (not the VPS):
# Blocks all incoming connections from other tailnet devices to this
 ↪  machine
tailscale set --shields-up

# To allow incoming connections again later:
tailscale set --shields-up=false
```

Это предотвращает подключение других устройств вашей tailnet к вашему ноутбуку — уменьшая поверхность атаки в случае компрометации другого устройства tailnet.

12.4) Межсетевой экран UFW для VPS только с Tailscale `[Intermediate]`

Если вы подключаетесь к VPS **только** через Tailscale (без публичного SSH), ограничьте межсетевой экран интерфейсом `tailscale0`.

> **Перед выполнением этих команд:** Убедитесь, что Tailscale подключён (`tailscale status`) и вы можете достучаться до VPS по его tailnet IP (`ping 100.x.y.z`). Если Tailscale не работает в момент включения UFW, вы потеряете доступ. Веб-консоль вашего провайдера (Hetzner Console, вкладка «Access» в DigitalOcean, «EC2 Instance Connect» в AWS) — это ваш путь восстановления в случае блокировки.

```
# Block everything from the public internet by default
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Only allow traffic arriving through the Tailscale VPN tunnel
# "tailscale0" is the virtual network interface Tailscale creates
sudo ufw allow in on tailscale0

# Activate the firewall (must set defaults before enabling)
sudo ufw enable
# Review the rules — you should see "ALLOW IN" only for tailscale0
sudo ufw status verbose
```

Это означает, что порт 22 доступен только через tailnet, а не из публичного интернета. Облачный межсетевой экран вашего провайдера также должен запрещать публичный SSH.

12.5) Настройка Tailscale Serve (HTTPS только для tailnet) [Intermediate]

`tailscale serve` открывает локальный порт для вашей tailnet с автоматическим TLS — certbot не нужен:

```
# Expose your local gateway (port 18789) as HTTPS on your private tailnet
# --bg runs it in the background; --https=443 means it listens on standard
 ↪  HTTPS port
sudo tailscale serve --bg --https=443 127.0.0.1:18789

# See what's currently being served
tailscale serve status

# Remove the serve config (stops exposing the port)
tailscale serve reset
```

Tailscale автоматически обеспечивает выпуск TLS-сертификатов. Доступ по адресу `https://<machine-name>.<tailnet>.ts.net/`.

Настройка OpenClaw:

```
{
  "gateway": {
    "bind": "loopback",
    "tailscale": { "mode": "serve" },
    "auth": { "allowTailscale": true }
  }
}
```

Или через CLI: `openclaw gateway --tailscale serve`

12.6) Funnel (публичный интернет) — используйте с осторожностью [Advanced]

Funnel открывает ваш сервис для **публичного интернета**, а не только для вашей tailnet:

```
# Make the gateway accessible from the PUBLIC INTERNET (not just your
 ↪  tailnet)
# WARNING: anyone on the internet can reach this — use strong auth
sudo tailscale funnel --https=443 127.0.0.1:18789
```

Порты Funnel ограничены значениями 443, 8443 или 10000.

Предупреждение: При использовании Funnel вы **должны** установить `gateway.auth.mode: "password"` — заголовки идентификации Tailscale НЕ доступны для запросов из публичного интернета. Избегайте использования Funnel для эндпоинтов управления браузером.

12.7) Усиление ACL (опционально) [Advanced]

Ограничьте доступ к tailnet, чтобы каждый пользователь мог подключаться только к своим устройствам. Настройте это в редакторе ACL консоли администрирования Tailscale:

```
{
  "acls": [
    {
      "action": "accept",
      "src": ["autogroup:member"],
      "dst": ["autogroup:self:*"]
    }
  ]
}
```

Это паттерн «пользователи имеют доступ к своим устройствам» — предотвращает доступ других участников tailnet к вашему VPS.

Справочник по конфигурации

Ключ	Значение	Назначение
`gateway.bind`	`"loopback"`	Шлюз слушает только на 127.0.0.1
`gateway.tailscale.mode`	`"serve"` / `"funnel"` / `"off"`	Уровень доступности
`gateway.auth.allowTailscale`	`true`	Принимать заголовки идентификации Tailscale
`gateway.tailscale.resetOnExit`	`true`	Очистка конфигурации Serve при завершении

Замечания по безопасности

- **Serve** = только tailnet (рекомендуется). **Funnel** = публичный интернет (требует `auth.mode: "password"`).
- `allowTailscale: true` позволяет запросам через прокси Tailscale автоматически аутентифицироваться через заголовки идентификации — ручной ввод токена не нужен.
- Избегайте Funnel для эндпоинтов управления браузером.
- Tailscale обеспечивает TLS автоматически — certbot не нужен.

Предварительные требования: Tailscale CLI установлен, аутентифицирован, HTTPS включён в вашей tailnet (для Serve).

Документация: gateway/tailscale, gateway/security

---

13) Усиление обратного прокси (nginx) [Advanced]

> Пропустите этот раздел, если вы выбрали Tailscale Serve (раздел 12) — Tailscale обеспечивает TLS и не требует обратного прокси.

OpenClaw **не имеет встроенного ограничения частоты запросов** (#8594) и добавляет заголовки безопасности только для эндпоинтов Control UI. Обратный прокси закрывает оба пробела.

Установка nginx

```
# Install the nginx web server (acts as a middleman between the internet
 ↪  and OpenClaw)
sudo apt install nginx
```

Полная усиленная конфигурация

Создайте `/etc/nginx/sites-available/openclaw`:

```
# Rate limiting zones
limit_req_zone $binary_remote_addr zone=general:10m rate=10r/s;
limit_req_zone $binary_remote_addr zone=auth:10m    rate=3r/s;

server {
    listen 443 ssl http2;
    server_name your-domain.com;

    # TLS (certbot fills these in — see section 14)
    ssl_certificate
 ↪  /etc/letsencrypt/live/your-domain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/your-domain.com/privkey.pem;

    # Security headers
    add_header Strict-Transport-Security "max-age=63072000;
 ↪  includeSubDomains; preload" always;
    add_header X-Content-Type-Options    "nosniff" always;
    add_header X-Frame-Options           "DENY" always;
    add_header Referrer-Policy           "no-referrer" always;
    add_header Permissions-Policy        "camera=(), microphone=(),
 ↪  geolocation=()" always;
    add_header Content-Security-Policy   "default-src 'self';
 ↪  frame-ancestors 'none'" always;

    # Request body limit (mitigates large-payload DoS)
    client_max_body_size 10m;

    # General rate limit
    limit_req zone=general burst=20 nodelay;

    # Stricter rate limit for auth endpoints
    location /api/auth {
        limit_req zone=auth burst=5 nodelay;
        proxy_pass http://127.0.0.1:18789;
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    # Main proxy with WebSocket support
    location / {
        proxy_pass http://127.0.0.1:18789;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # WebSocket upgrade
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_read_timeout 86400s;
    }

    # Optional: IP allowlist (uncomment and replace with your IP)
    # allow YOUR_IP;
    # deny all;
}

# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name your-domain.com;
    return 301 https://$host$request_uri;
}
```

Активация сайта:

```
# Create a symbolic link to activate the config (nginx reads from
 ↪  sites-enabled/)
sudo ln -s /etc/nginx/sites-available/openclaw /etc/nginx/sites-enabled/
# Remove the default "Welcome to nginx" site
sudo rm /etc/nginx/sites-enabled/default
```

Примечание: Выполняйте `sudo nginx -t && sudo systemctl reload nginx` только **после** получения TLS-сертификата в разделе 14. Приведённая выше конфигурация ссылается на пути к сертификатам, которых не будет до тех пор, пока certbot их не создаст.

Настройка доверенных прокси (обязательно)

При использовании обратного прокси вы **должны** указать OpenClaw доверять заголовку `X-Forwarded-For` от прокси. Добавьте в `openclaw.json`:

```
{
  "gateway": {
    "trustedProxies": ["127.0.0.1"]
  }
}
```

Без этого ограничение частоты на основе IP и логирование будут видеть IP прокси вместо реального IP клиента.

Перекрёстная ссылка: Чек-лист усиления, раздел 11

Документация: gateway/security, nginx rate limiting

---

14) Терминация TLS [Advanced]

> Пропустите этот раздел, если вы выбрали Tailscale Serve (раздел 12) — TLS обеспечивается автоматически.

OpenClaw не устанавливает заголовки HSTS. Обратный прокси с TLS от Let's Encrypt решает эту проблему.

Установка certbot

```
# Install certbot (the Let's Encrypt client) and its nginx plugin
sudo apt install certbot python3-certbot-nginx
```

Получение сертификата

```
# Request a free TLS certificate for your domain and auto-configure nginx
# You'll be asked for an email address and to agree to terms of service
sudo certbot --nginx -d your-domain.com
```

Certbot автоматически обновит вашу конфигурацию nginx, добавив пути к сертификатам.

Автоматическое продление

```
# Set up a cron job that checks for certificate renewal twice daily (at
 ↪  midnight and noon)
# The random sleep prevents millions of servers hitting Let's Encrypt at
 ↪  the same second
SLEEPTIME=$(awk 'BEGIN{srand(); print int(rand()*(3600+1))}')
echo "0 0,12 * * * root sleep $SLEEPTIME && certbot renew -q" | sudo tee
 ↪  -a /etc/crontab > /dev/null
```

Проверка продления выполняется дважды в день с случайной задержкой для предотвращения пиковых нагрузок на Let's Encrypt.

Документация: certbot, gateway/tailscale

---

15) Зашифрованное хранилище секретов [Advanced]

OpenClaw хранит **все секреты** (API-ключи, OAuth-токены, токены шлюза) в **открытом виде** на диске. Права доступа к файловой системе (`0600`/`0700`) — единственная защита. Встроенного шифрования данных в состоянии покоя не существует.

Вариант A: Зашифрованный раздел LUKS

```
# Install the disk encryption tools
sudo apt install cryptsetup

# Set up encryption on an empty disk/partition — DESTROYS ALL DATA on
 ↪  /dev/sdX
# You'll be asked to create a passphrase (this unlocks the disk on reboot)
sudo cryptsetup luksFormat /dev/sdX
# Unlock the encrypted volume (prompts for your passphrase)
sudo cryptsetup open /dev/sdX openclaw-vault
# Create a filesystem inside the encrypted volume
sudo mkfs.ext4 /dev/mapper/openclaw-vault

# Mount it as the OpenClaw data directory
sudo mount /dev/mapper/openclaw-vault /home/openclaw/.openclaw
# Give the openclaw user ownership
sudo chown openclaw:openclaw /home/openclaw/.openclaw
```

Добавьте зависимость монтирования в systemd, чтобы сервис OpenClaw ожидал расшифровки:

```
# In /etc/systemd/system/openclaw-gateway.service
[Unit]
RequiresMountsFor=/home/openclaw/.openclaw
```

Примечание: Требует ручной разблокировки при перезагрузке (или файла ключа, что жертвует безопасностью ради удобства).

Вариант B: Шифрование на уровне провайдера

Провайдер	Шифрование данных в состоянии покоя
DigitalOcean	Блочные тома зашифрованы по умолчанию
AWS	Включите шифрование EBS на томе
Hetzner	Включите LUKS при подготовке инстанса

Что это защищает

- Кража диска, списанный VPS, утечка снапшотов.

Что это НЕ защищает

- Память работающего процесса, аутентифицированный SSH-доступ.

Документация: gateway/security (подтверждает отсутствие встроенного шифрования)

---

16) Усиление Docker-развёртывания [Advanced]

По умолчанию `docker-compose.yml` привязывается к `0.0.0.0` (режим lan) и не имеет включённых параметров безопасности.

Переопределение адреса привязки

Добавьте в ваш файл `.env`:

```
OPENCLAW_GATEWAY_BIND=loopback
```

Это переопределяет значение по умолчанию `${OPENCLAW_GATEWAY_BIND:-lan}` в `docker-compose.yml`.

Overlay безопасности Docker Compose

Создайте `docker-compose.override.yml` или добавьте в ваш существующий compose-файл:

```
services:
  openclaw-gateway:
    security_opt:
      - no-new-privileges:true
    cap_drop:
      - ALL
    cap_add:
      - CHOWN
      - SETUID
      - SETGID
    read_only: true
    tmpfs:
      - /tmp
      - /home/node/.openclaw/workspace
    pids_limit: 256
    mem_limit: 1g
```

Примечание: Флаг Chrome `--no-sandbox` обязателен в контейнерах (по умолчанию в Dockerfile) и не может быть отключён.

Конфигурация Docker-песочницы OpenClaw

Эти настройки управляют Docker-песочницей, которую OpenClaw создаёт для сессий агента. Добавьте в `openclaw.json`:

```
{
  "agents": {
    "defaults": {
      "sandbox": {
        "mode": "all",
        "docker": {
          "network": "none",
          "readOnlyRoot": true,
          "capDrop": ["ALL"],
          "pidsLimit": 100,
          "memory": "512m"
        }
      }
    }
  }
}
```

Оговорка: `readOnlyRoot`, `capDrop`, `pidsLimit` и `memory` определены в исходном коде (`src/config/types.sandbox.ts`), но ещё не в официальной документации. Они работают (протестированы в `src/config/config.sandbox-docker.test.ts`), но могут измениться между релизами.

Документация: Docker Compose services reference, gateway/sandboxing

---

17) Усиление конфигурации OpenClaw [Advanced]

Существует несколько параметров конфигурации, связанных с безопасностью, которые не включены по умолчанию. Этот раздел объединяет рекомендуемые настройки для усиленного VPS-развёртывания.

Рекомендуемый `openclaw.json`

```
{
  "gateway": {
    "bind": "loopback",
    "auth": {
      "mode": "token"
    },
    "trustedProxies": ["127.0.0.1"],
    "controlUi": {
      "dangerouslyDisableDeviceAuth": false
    }
  },
  "browser": {
    "evaluateEnabled": false
  },
```

```
  "plugins": {
    "enabled": false
  },
  "agents": {
    "defaults": {
      "sandbox": {
        "mode": "all",
        "workspaceAccess": "ro"
      }
    }
  },
  "logging": {
    "redactSensitive": "tools"
  }
}
```

Описание настроек

Настройка	Назначение
`gateway.bind: "loopback"`	Слушать только на 127.0.0.1 — доступ через SSH-туннель или Tailscale
`gateway.auth.mode: "token"`	Требовать bearer-токен для всех запросов (документация)
`gateway.trustedProxies: ["127.0.0.1"]`	Обязательно при использовании обратного прокси — предотвращает подделку X-Forwarded-For (документация)
`dangerouslyDisableDeviceAuth: false`	Предотвращает скрытое ослабление аутентификации
`browser.evaluateEnabled: false`	Отключает произвольное выполнение JS в контексте браузера (`src/config/types.browser.ts:18`)
`plugins.enabled: false`	Предотвращает обход аутентификации HTTP-маршрутов плагинов (#8512)
`agents.defaults.sandbox.mode: "all"`	Все сессии запускаются в Docker-песочнице (документация)
`agents.defaults.sandbox.workspaceAccess: "ro"`	Рабочее пространство только для чтения предотвращает подмену файлов (документация)
`logging.redactSensitive: "tools"`	Редактирование секретов в логах вывода инструментов (документация)

Токен шлюза

Устанавливается через переменную окружения (>= 32 символа):

```
# Generate a new random 64-character hex token and set it as an environment
 ↪  variable
export OPENCLAW_GATEWAY_TOKEN="$(openssl rand -hex 32)"
```

Проверка

После применения изменений выполните:

```
openclaw security audit --deep
```

Документация: gateway/security, gateway/sandboxing

---

18) Ротация токена шлюза [Advanced]

OpenClaw не имеет автоматической ротации токенов шлюза. Простой cron-скрипт восполняет этот пробел.

Скрипт ротации

Скрипт зависит от способа установки OpenClaw. По умолчанию `openclaw onboard --install-daemon` хранит токен в `~/.openclaw/openclaw.json` и запускается как **пользовательский systemd-сервис** (`openclaw-gateway.service`). Развёртывание DO 1-Click использует системный сервис с `/opt/clawdbot.env`.

Установка по умолчанию (пользовательский systemd-сервис — большинство ручных VPS-установок):

Создайте `/usr/local/bin/rotate-openclaw-token.sh`:

```
#!/bin/bash
# Rotation script for default OpenClaw installs
# Token location: ~/.openclaw/openclaw.json
# Service: systemd user unit "openclaw-gateway.service"

OPENCLAW_USER="openclaw"
OPENCLAW_HOME="/home/${OPENCLAW_USER}"
CONFIG="${OPENCLAW_HOME}/.openclaw/openclaw.json"

# Generate a new random token
NEW_TOKEN=$(openssl rand -hex 32)

# Replace the token in openclaw.json using jq (a JSON command-line
 ↪  processor)
# Writes to a temp file first to avoid corrupting the config if interrupted
jq --arg t "$NEW_TOKEN" '.gateway.auth.token = $t' "$CONFIG" >
 ↪  "${CONFIG}.tmp" \
  && mv "${CONFIG}.tmp" "$CONFIG" \
  && chown "${OPENCLAW_USER}:${OPENCLAW_USER}" "$CONFIG"

# Restart the gateway so it picks up the new token
sudo -u "$OPENCLAW_USER" XDG_RUNTIME_DIR="/run/user/$(id -u
 ↪  $OPENCLAW_USER)" \
  systemctl --user restart openclaw-gateway

# Log the rotation for audit trail
echo "$(date): Token rotated" >> /var/log/openclaw-token-rotation.log
```

Установка DO 1-Click (системный сервис с env-файлом):

```
#!/bin/bash
# Rotation script for DigitalOcean 1-Click installs
# Token location: /opt/clawdbot.env, service name: "moltbot"

NEW_TOKEN=$(openssl rand -hex 32)
# Find the OPENCLAW_GATEWAY_TOKEN line in the env file and replace its
 ↪  value
sed -i "s/^OPENCLAW_GATEWAY_TOKEN=.*/OPENCLAW_GATEWAY_TOKEN=${NEW_TOKEN}/"
 ↪  /opt/clawdbot.env
# Restart the service so it reads the new token
systemctl restart moltbot
echo "$(date): Token rotated" >> /var/log/openclaw-token-rotation.log
```

Сделайте скрипт исполняемым:

```
# Mark the script as executable so cron can run it
sudo chmod +x /usr/local/bin/rotate-openclaw-token.sh
```

Предварительное требование: Скрипт для установки по умолчанию использует `jq` — установите с помощью `sudo apt install jq`, если отсутствует.

Планирование ежемесячной ротации

```
# Add a monthly cron job: runs at 3 AM on the 1st of each month as root
echo "0 3 1 * * root /usr/local/bin/rotate-openclaw-token.sh" | sudo tee
 ↪  -a /etc/crontab > /dev/null
```

Запускается в 3 часа ночи 1-го числа каждого месяца.

Важно: После ротации обновите все сохранённые URL панели управления или конфигурации SSH-туннелей, содержащие старый токен.

Документация: gateway/security (документация по аутентификации токенов)

Чек-лист безопасности (VPS)

На основе VibeProof Security Guide (используется устаревшее название «Moltbot»).

Базовое усиление

- ☐ Аутентификация SSH по паролю отключена (`PasswordAuthentication no`)
- ☐ Вход SSH под root отключён (`PermitRootLogin no`)
- ☐ Конфигурация SSH проверена с помощью `sshd -t` перед перезагрузкой
- ☐ Файл подкачки настроен (предотвращает завершение процессов из-за OOM)
- ☐ Рассылка mDNS/Bonjour отключена (`OPENCLAW_DISABLE_BONJOUR=1`)
- ☐ Синхронизация времени NTP активна (chrony или systemd-timesyncd)
- ☐ Расписание резервного копирования конфигурации активно

Сеть

- ☐ Входящие правила группы безопасности ограничены SSH только с вашего IP
- ☐ Порт шлюза 18789 НЕ является публичным
- ☐ Межсетевой экран хоста (UFW) включён
- ☐ Fail2ban активен для защиты SSH

Аутентификация и доступ

- ☐ Токен аутентификации шлюза установлен
- ☐ Политика личных сообщений установлена на `allowlist` или `pairing`
- ☐ Только одобренные user ID могут инициировать действия

Безопасность выполнения

- ☐ Docker-песочница включена для инструментов выполнения
- ☐ Песочница имеет `network: none` или строгую изоляцию
- ☐ Опасные шаблоны команд заблокированы (`rm -rf`, `curl | bash` и т.д.)
- ☐ Инструменты ограничены минимально необходимым набором

Секреты

- ☐ Секреты хранятся в переменных окружения (не в истории команд)
- ☐ Для файлов с конфиденциальными данными установлен `chmod 600`
- ☐ История команд защищена (`HISTCONTROL=ignoreboth`)
- ☐ `~/.openclaw/` на зашифрованном томе (LUKS или на уровне провайдера)

Обслуживание системы

- ☐ Автоматические обновления безопасности включены
- ☐ Установлен Node.js 22.12.0+
- ☐ Ограничения ресурсов systemd настроены

Наблюдаемость

- ☐ Логирование сессий включено
- ☐ Ротация логов активна (`/var/log/openclaw/`)
- ☐ Привычка еженедельного обзора

Tailscale (при использовании пути Tailscale — раздел 12)

- ☐ Tailscale установлен и аутентифицирован на VPS
- ☐ Tailscale установлен на всех клиентских устройствах (тот же провайдер идентификации)
- ☐ Настроен `tailscale.mode: "serve"` (только tailnet)
- ☐ Установлен `gateway.auth.allowTailscale: true`
- ☐ Shields-up включён на персональных устройствах (`tailscale set --shields-up`)

- ☐ UFW ограничен интерфейсом `tailscale0` (если только Tailscale)
- ☐ Funnel НЕ включён, если явно не требуется (требует аутентификации по паролю)
- ☐ Политика ACL проверена в консоли администрирования

Обратный прокси и TLS (при использовании пути nginx — разделы 13–14)

- ☐ Обратный прокси (nginx/Caddy) перед шлюзом
- ☐ Ограничение частоты запросов настроено (общее + эндпоинты аутентификации)
- ☐ Заголовки безопасности установлены (HSTS, CSP, X-Frame-Options, X-Content-Type-Options, Referrer-Policy)
- ☐ Терминация TLS с действительным сертификатом
- ☐ `gateway.trustedProxies` настроен

Зашифрованное хранилище (раздел 15)

- ☐ `~/.openclaw/` на зашифрованном томе (LUKS или на уровне провайдера)
- ☐ Права доступа к расшифровкам сессий 600 (не 644)

Усиление Docker (для Docker-развёртываний — раздел 16)

- ☐ `OPENCLAW_GATEWAY_BIND=loopback` в `.env`
- ☐ `cap_drop: ALL` с минимальным `cap_add`
- ☐ Параметр безопасности `no-new-privileges` включён
- ☐ `read_only: true` с tmpfs для записываемых путей
- ☐ Ограничения ресурсов (память, PIDs) настроены

Конфигурация OpenClaw (раздел 17)

- ☐ `browser.evaluateEnabled: false`
- ☐ Плагины отключены или добавлены в список разрешённых
- ☐ Режим песочницы `"all"` для агентов, обрабатывающих ненадёжный ввод
- ☐ Токен шлюза >= 32 символа
- ☐ Расписание ротации токенов активно

> **См. также:** Атаки инъекцией промптов — 27 примеров атак. VPS-развёртывания с доступом из интернета имеют большую поверхность атаки через инъекцию промптов, чем локальные установки.

Руководство по развёртыванию: Cloudflare Moltworker (бессерверный)

> **Примечание:** Это руководство для OpenClaw (ранее Moltbot/Clawdbot). Moltworker — экспериментальное бессерверное развёртывание, а не официальный продукт Cloudflare.

Что такое Moltworker? (простым языком)

Moltworker — это Gateway, работающий внутри инфраструктуры Cloudflare вместо оборудования, которое вы управляете. Вместо Mac mini в шкафу или VPS, к которому вы подключаетесь по SSH, Gateway работает как **Cloudflare Worker** внутри контейнерной среды **Sandbox SDK**.

Почему вы можете это выбрать:

- Нет оборудования для обслуживания, обновления или поддержания в рабочем состоянии
- Автоматическое масштабирование и географическое распределение через edge-сеть Cloudflare
- Встроенная изоляция — каждое выполнение запускается в изолированном контейнере
- Оплата по факту использования (нет расходов на простой сервера)

Почему вы можете это не выбрать:

- Статус экспериментальной разработки — ещё не усилен для продакшна
- Требуется платный план Cloudflare Workers (минимум $5/месяц)
- Некоторые инструменты (доступ к локальным файлам, постоянные сессии браузера) работают иначе
- Меньше контроля над средой выполнения

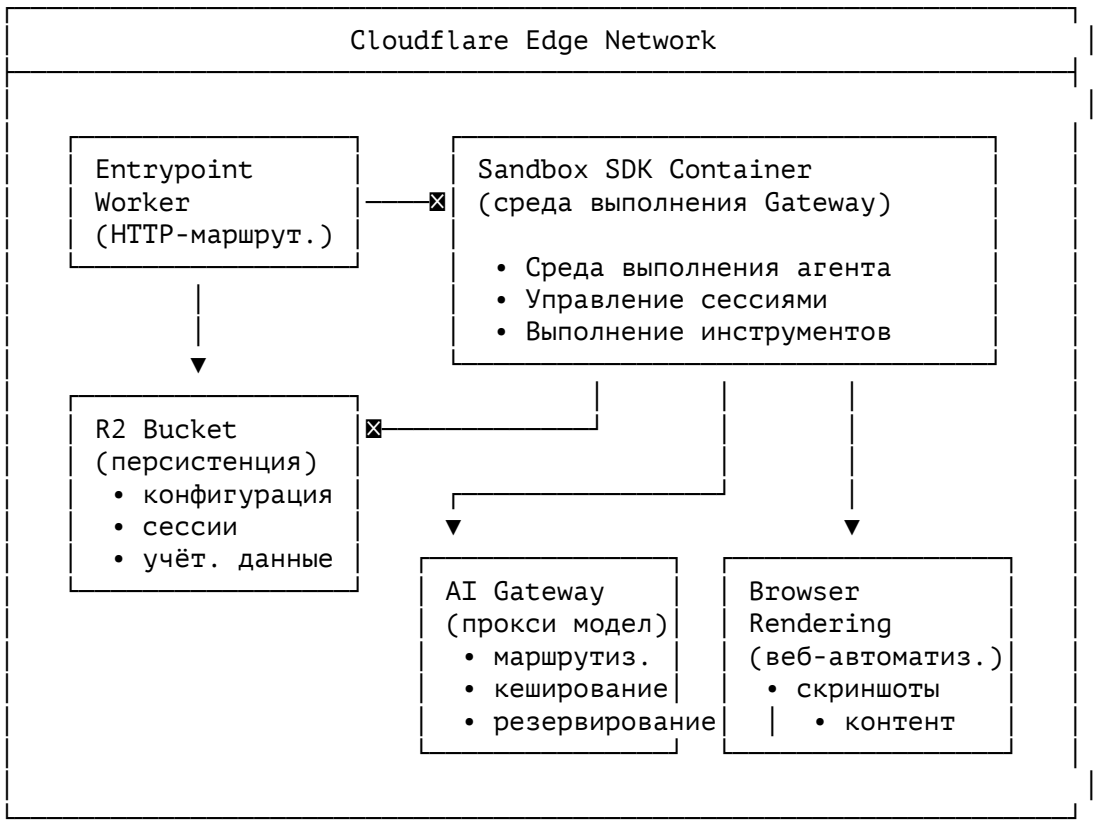
Перед началом использования: прочтите это

Прежде чем начать использовать OpenClaw ежедневно, ознакомьтесь с операционными подводными камнями реальных пользователей:

- **Правило 60% успеха** — задачи с более чем 10 шагами проваливаются в 40% случаев из-за дрейфа контекста
- **Неоднозначность «черновик vs отправка»** — агенты могут интерпретировать «draft» как «создать и отправить»
- **Утечка профиля браузера** — использование повседневного профиля Chrome даёт агенту доступ ко ВСЕМ вашим активным сессиям
- **Ловушка бездействия** — длинные сессии приводят к зависанию агента или потере контекста
- **Стоимость постоянной работы** — работа 24/7 стоит дороже, чем ожидается (473 запроса/день = $847/месяц в одном случае)
- **Специфика Moltworker** — отсутствие фильтрации исходящего трафика означает, что успешная инъекция промптов может эксфильтрировать данные на любой сервер

Техническая архитектура

Moltworker использует пять сервисов Cloudflare, работающих совместно:



Ответственности компонентов

Компонент	Что он делает
Entrypoint Worker	Принимает HTTP-запросы (вебхуки, вызовы API), маршрутизирует в песочницу
Sandbox SDK Container	Выполняет среду Gateway с полным окружением Node.js
R2 Bucket	Хранит конфигурацию, транскрипты сессий и учётные данные (шифрование при хранении)
AI Gateway	Проксирует вызовы API моделей с кешированием, ограничением скорости и резервной маршрутизацией
Browser Rendering	Предоставляет headless Chromium для инструментов веб-автоматизации

Поток архитектуры

1. **Входящие запросы** (вебхуки от Telegram/Discord, вызовы API) поступают в Entrypoint Worker
2. **Entrypoint Worker** аутентифицирует запрос и маршрутизирует его в контейнер Sandbox
3. **Sandbox SDK Container** выполняет полную среду Gateway, обрабатывая сообщения и запуская инструменты
4. **R2 Bucket** обеспечивает постоянное хранилище для конфигурации, сессий и учётных данных (переживает перезапуски контейнера)
5. **AI Gateway** маршрутизирует вызовы API моделей с кешированием, ограничением скорости и резервированием провайдеров
6. **Browser Rendering** обрабатывает инструменты веб-автоматизации (скриншоты и извлечение контента)

Сервисы платформы Cloudflare

Sandbox SDK (бета)

Простым языком: Представьте полноценный Linux-компьютер в облаке, запускающийся по запросу. Когда кто-то отправляет сообщение OpenClaw, Cloudflare создаёт свежий изолированный контейнер только для этого запроса.

Технические детали:

- Построен на **Cloudflare Containers** (бессерверная среда контейнеров) + **Durable Objects** (координация с состоянием)
- Полное окружение Linux с Python 3.x, Node.js, pip, npm
- Пакет `@cloudflare/sandbox` для npm предоставляет TypeScript API:

```
import { getSandbox } from '@cloudflare/sandbox';
const sandbox = getSandbox(env.Sandbox, 'session-123');
await sandbox.exec('python script.py');
await sandbox.writeFile('/workspace/config.json', data);
```

- Ключевые методы: `exec()`, `writeFile()`, `readFile()`, `startProcess()`, `exposePort()`, `gitCheckout()`
- **Ленивый запуск:** Контейнер создаётся только при первой операции
- **Спящий режим:** `sleepAfter: "10m"` — гибернация контейнера при неактивности
- **Требует платный план Workers** ($5/мес минимум)

Документация: Sandbox SDK

R2 Object Storage

Простым языком: R2 — версия Amazon S3 от Cloudflare для хранения файлов в облаке. Главное преимущество — **нулевые сборы за исходящий трафик**: вы никогда не платите за скачивание данных, в отличие от AWS/GCP.

Технические детали:

- **S3-совместимый API**
- **99,999999999% надёжности** (11 девяток)
- **Строгая консистентность** — чтение немедленно видит последние записи
- API привязки Workers:

```
// Запись
await env.MY_BUCKET.put('sessions/user-123.json',
 ↪  JSON.stringify(data));
// Чтение
const object = await env.MY_BUCKET.get('sessions/user-123.json');
const data = await object.json();
```

- **Бесплатный уровень:** 10 ГБ хранилища, 1M операций записи, 10M операций чтения
- **Платный:** $0.015/ГБ-мес хранилища, $4.50/M записей, $0.36/M чтений, нулевой исходящий трафик

Документация: Cloudflare R2

AI Gateway

Простым языком: AI Gateway стоит между вашим приложением и провайдерами ИИ (Anthropic/OpenAI). Вместо прямого вызова Claude вы вызываете AI Gateway, который вызывает Claude за вас. Преимущества: кеширование, ограничение скорости, резервирование, аналитика.

Технические детали:

- **Интеграция в одну строку:** смена базового URL API
- **Поддержка 20+ провайдеров:** Anthropic, OpenAI, Azure OpenAI, Amazon Bedrock, Google Vertex AI, Cohere, Hugging Face, Mistral и др.
- **Кеширование:** обслуживание идентичных запросов из edge-кеша — снижение задержки до 90%
- **Ограничение скорости:** скользящее или фиксированное окно
- **Динамическая маршрутизация:** резервирование моделей, A/B-тестирование, маршрутизация по геолокации
- **Доступен на всех планах** (Free и Paid)

Документация: AI Gateway

Browser Rendering

Простым языком: Browser Rendering предоставляет headless-браузер Chrome в сети Cloudflare. Ваш код может посещать сайты, делать скриншоты, генерировать PDF или извлекать контент.

Технические детали:

- **Headless Chromium** на глобальной edge-сети Cloudflare
- **Два метода интеграции:**
  1. **REST API** (простой): `/screenshot`, `/pdf`, `/content`, `/markdown`, `/scrape`, `/json`, `/links`
  2. **Workers Bindings** (полная автоматизация): Puppeteer, Playwright, Stagehand
- **Лимиты:** Free: 10 мин/день; Paid: 10 ч/мес, затем $0.09/час
- **Доступен на Free и Paid планах**

Документация: Browser Rendering

Durable Objects

Простым языком: Durable Objects — маленькие базы данных, живущие близко к вашим пользователям. Каждый объект имеет уникальный ID, собственное хранилище и может координировать запросы.

Технические детали:

- **Строго консистентная, однопоточная координация**
- Частное SQLite-хранилище или key-value хранилище
- Поддержка WebSocket, планирование алармов
- Объекты мигрируют ближе к обращающимся пользователям
- Используется внутри Sandbox SDK, R2 Metadata Service, Browser Rendering

Документация: Durable Objects

Cloudflare Access (Zero Trust)

Простым языком: Cloudflare Access — как охранник для ваших веб-приложений. Вместо публикации панели администратора с паролем, Access требует подтверждения личности перед показом страницы входа.

Технические детали:

- **Прокси с поддержкой идентификации**
- **Множество провайдеров идентификации:** Email OTP, Google, GitHub, Okta, Azure AD, SAML и др.
- **Политики доступа:** на основе домена email, группы, геолокации, состояния устройства
- **Защищённые маршруты в Moltworker:** `/_admin/`, `/api/*`, `/debug/*`

Документация: Cloudflare Access

Предварительные требования

1. **Аккаунт Cloudflare** с платным планом Workers ($5/мес минимум)
2. **API-ключи** для провайдера(ов) моделей (Anthropic, OpenAI и др.)
3. **Wrangler CLI** установлен локально (`npm install -g wrangler`)
4. **Git** для клонирования и развёртывания

```
# Установить wrangler при необходимости
npm install -g wrangler

# Войти в Cloudflare
wrangler login

# Проверить аккаунт
wrangler whoami
```

Пошаговая настройка

1) Клонируйте репозиторий Moltworker

```
git clone https://github.com/cloudflare/moltworker.git
cd moltworker
npm install
```

2) Создайте необходимые ресурсы Cloudflare

```
# Создать R2-бакет для персистенции
wrangler r2 bucket create moltworker-state

# Создать KV namespace для быстрых поисков (опционально)
wrangler kv:namespace create MOLTWORKER_KV
```

3) Настройте wrangler.toml

```
name = "moltworker"
main = "src/index.ts"
compatibility_date = "2024-01-01"

[vars]
GATEWAY_MODE = "serverless"

[[r2_buckets]]
binding = "STATE_BUCKET"
bucket_name = "moltworker-state"
```

4) Установите секреты

Секрет	Обязателен	Назначение
ANTHROPIC_API_KEY или AI_GATEWAY_API_KEY	Да	Доступ к LLM
MOLTBOT_GATEWAY_TOKEN	Да	Аутентификация Control UI
CF_ACCESS_TEAM_DOMAIN	Рекомендуется	Домен команды Cloudflare Access
CF_ACCESS_AUD	Рекомендуется	Тег аудитории Cloudflare Access
TELEGRAM_BOT_TOKEN	Опционально	Канал Telegram
DISCORD_BOT_TOKEN	Опционально	Канал Discord

```
# Обязательно: API-ключ провайдера модели
wrangler secret put ANTHROPIC_API_KEY

# Обязательно: токен аутентификации gateway
wrangler secret put MOLTBOT_GATEWAY_TOKEN
```

```
# Рекомендуется: защита Cloudflare Access
wrangler secret put CF_ACCESS_TEAM_DOMAIN
wrangler secret put CF_ACCESS_AUD
```

5) Разверните

```
wrangler deploy
```

6) Проверьте развёртывание

```
# Проверка здоровья
curl https://moltworker.your-subdomain.workers.dev/health

# Статус (требует аутентификации)
curl -H "Authorization: Bearer YOUR_GATEWAY_AUTH_TOKEN" \
  https://moltworker.your-subdomain.workers.dev/api/status
```

Обзор стоимости

Сервис	Бесплатный уровень	Платная стоимость	Примечания
Workers Paid	Н/Д	$5/мес базовая	Обязателен для Sandbox SDK
Containers/Sandbox	Включено в Workers Paid	25 ГиБ-ч памяти, 375 vCPU-мин, 200 ГБ-ч диска/мес	Масштабируется до нуля при простое
R2	10 ГБ, 1М записей, 10М чтений	$0.015/ГБ, $4.50/М записей, $0.36/М чтений	Нулевой исходящий трафик
AI Gateway	Включено	Включено	Все планы
Browser Rendering	10 мин/день	10 ч/мес, затем $0.09/час	—
Cloudflare Access	50 пользователей бесплатно	$3/пользователь/мес (сверх бесплатного)	Опционально, рекомендуется

Оценка для лёгкого персонального использования: $5-15/месяц

Уровни аутентификации

1) Cloudflare Access (рекомендуется)

Защищает административные маршруты с проверкой идентичности.

2) Токен Gateway

Обязателен для доступа к Control UI, передаётся через параметр запроса.

3) Привязка устройств

Новые устройства требуют явного одобрения.

Стратегия персистенции

- **Интервал резервного копирования:** каждые 5 минут через cron-задачу
- **Ручной запуск:** доступен в Admin UI
- **Риск потери данных:** перезапуск контейнера без настройки R2 ведёт к потере данных

Заметки по производительности

- **Холодный старт:** 1-2 минуты для начальной раскрутки контейнера
- **Последующие запросы:** значительно быстрее пока контейнер «тёплый»
- **Список устройств:** задержка 10-15 секунд
- **WebSocket:** ограничен в локальной разработке; полная функциональность в продакшне

Вопросы безопасности

Что предоставляет Cloudflare

- **Изоляция:** каждый запрос выполняется в свежем V8-изоляте или контейнере песочницы
- **Шифрование:** TLS для всего трафика, шифрование при хранении для R2
- **Интеграция Zero Trust:** возможность требовать Cloudflare Access для аутентификации
- **Защита от DDoS:** встроена в edge-сеть

За что отвечаете вы

- **Безопасность учётных данных:** API-ключи хранятся как секреты Worker (зашифрованы)
- **Аутентификация Gateway:** установите надёжный `GATEWAY_AUTH_TOKEN`
- **Контроль доступа:** настройте привязку/списки разрешённых как в других развёртываниях
- **Мониторинг:** проверяйте логи в панели Cloudflare

Отличия от самостоятельного хостинга

Аспект	Самостоятельный (Mac/VPS)	Moltworker
Граница доверия	Ваше оборудование	Инфраструктура Cloudflare
Хранение учётных данных	Локальная файловая система	Секреты Cloudflare + R2
Сетевая изоляция	Ваш файрвол	Edge-сеть Cloudflare
Изоляция выполнения	Docker-песочница (опционально)	Контейнер Sandbox SDK

Важно: Если вам необходимо, чтобы учётные данные никогда не покидали оборудование, которое вы контролируете, Moltworker — неподходящий выбор.

Сравнение: Mac mini vs VPS vs Moltworker

Аспект	Mac mini	VPS	Moltworker
Сложность настройки	Низкая	Средняя	Средняя
Текущее обслуживание	Среднее	Высокое	Низкое (управляемое)
Стоимость	Оборудование (~$500+)	$6-20/мес	$5-10/мес
Доступность	Зависит от питания/сети	Высокая (SLA провайдера)	Высокая (SLA Cloudflare)
Приватность	Максимальная (ваше оборудование)	Средняя	Ниже (инфраструктура Cloudflare)
Масштабирование	Фиксированная мощность	Ручное	Автоматическое
Поддержка инструментов	Полная	Полная	Ограниченная
Лучше всего для	Приватность, домашние пользователи	Постоянная работа, удалённый доступ	Минимум обслуживания

Ограничения

Статус экспериментальной разработки

Moltworker экспериментален. Он демонстрирует, что Gateway может работать бессерверно, но:

- Не все инструменты работают идентично (нет локальной файловой системы, нет постоянных сессий браузера)
- Долгие операции могут упереться в лимиты CPU Worker
- Доставка вебхуков некоторых каналов может требовать дополнительной настройки
- Не покрывается гарантиями стабильности OpenClaw
- Может измениться без предупреждения

Устранение неполадок

Worker возвращает ошибки 500

Проверьте логи в панели Cloudflare. Частые причины: отсутствующие секреты, ненастроенный R2-бакет, ошибки привязок в wrangler.toml.

Ошибка аутентификации Gateway

```
wrangler secret put GATEWAY_AUTH_TOKEN
curl -H "Authorization: Bearer YOUR_TOKEN" \
  https://moltworker.your-subdomain.workers.dev/api/status
```

Сессии не сохраняются

```
wrangler r2 object list moltworker-state
```

Вызовы API модели не работают

1. Проверьте, что API-ключ установлен: `wrangler secret list`
2. Проверьте логи AI Gateway (при использовании)
3. Протестируйте прямой вызов API для исключения проблем провайдера

Чек-лист безопасности (Moltworker)

Конфигурация Cloudflare

- ☐ Платный план Workers активен
- ☐ Worker развёрнут с последним кодом
- ☐ R2-бакет создан с приватным доступом
- ☐ Cloudflare Access защищает административные маршруты

Секреты

- ☐ `MOLTBOT_GATEWAY_TOKEN` установлен (надёжное случайное значение)
- ☐ `CF_ACCESS_TEAM_DOMAIN` и `CF_ACCESS_AUD` установлены
- ☐ API-ключи провайдеров моделей установлены как секреты
- ☐ Никаких секретов в wrangler.toml или закоммиченном коде

Контроль доступа

- ☐ Токен аутентификации Gateway обязателен для Control UI
- ☐ Cloudflare Access обязателен для административных маршрутов
- ☐ Политика ЛС — `allowlist` или `pairing`
- ☐ Привязка устройств включена

Персистенция

- ☐ R2-бакет настроен для сохранения состояния
- ☐ Cron-задача резервного копирования работает (каждые 5 минут)
- ☐ Ручное резервное копирование протестировано через Admin UI

Следующие шаги

После развёртывания:

1. **Подключите канал** — настройте Telegram, Discord или другой канал
2. **Настройте привязку** — убедитесь, что только вы можете взаимодействовать с ботом
3. **Протестируйте инструменты** — проверьте работу web fetch, browser rendering
4. **Мониторьте использование** — проверяйте панель Cloudflare на запросы и расходы

Для настройки каналов см.: https://docs.openclaw.ai/start/pairing

Руководство по развёртыванию: Docker Model Runner (локальный ИИ, нулевые затраты на API)

Примечание: Данное руководство предназначено для OpenClaw (ранее Moltbot/Clawdbot).

Содержание (Explain OpenClaw)

Цель: запустить OpenClaw с **локальными ИИ-моделями** через Docker Model Runner при **нулевых затратах на API** и **полной конфиденциальности**.

Эта конфигурация идеальна, когда:

- Вам необходима полная конфиденциальность (данные не покидают вашу машину)
- Вам необходимы нулевые текущие затраты на API
- У вас имеется производительное оборудование (Apple Silicon, NVIDIA GPU или AMD GPU)
- Вы хотите работать офлайн

Связанная официальная документация:

- https://docs.docker.com/desktop/features/model-runner/
- https://docs.openclaw.ai/gateway/local-models
- Docker Blog: Private Personal AI with Clawdbot + DMR
- Docker Blog: OpenCode + Model Runner for Private AI Coding

Что такое Docker Model Runner?

Docker Model Runner (DMR) — это встроенный инструмент Docker Desktop для локального запуска LLM. Он предоставляет:

- **OpenAI-совместимый API** по адресу `http://model-runner.docker.internal/v1`
- **Автоматическое управление ресурсами** — модели загружаются по запросу, выгружаются при простое
- **Несколько движков инференса** — llama.cpp (все платформы), vLLM (NVIDIA), Diffusers (генерация изображений)
- **Возможность отключения сбора данных**

Простым языком: Вместо отправки сообщений в Anthropic/OpenAI (с оплатой за каждый токен) вы запускаете локальную ИИ-модель на собственном компьютере. Docker Desktop берёт на себя всю сложность — загрузку моделей, GPU-ускорение и предоставление API, идентичного OpenAI.

Сценарий	Рекомендация

Когда использовать DMR, а когда — облачных провайдеров

Сценарий	Рекомендация
Максимальная конфиденциальность	**DMR** — данные не покидают вашу машину
Нулевые текущие затраты	**DMR** — только однократная загрузка модели
Лучшее качество модели	**Облако** — Claude Opus, GPT-4o всё ещё более производительны
Работа офлайн	**DMR** — работает без интернета после настройки
Ограниченное оборудование	**Облако** — нет требований к GPU/RAM
Быстрая настройка	**Облако** — достаточно добавить API-ключ
Продакшн/коммерческое использование	**Облако** — лучшие гарантии надёжности и качества

Итог: DMR отлично подходит для личного использования с упором на конфиденциальность, разработки/тестирования и обучения. Для продакшн-нагрузок, требующих ответов наивысшего качества, облачные провайдеры всё ещё имеют преимущество.

Системные требования

Docker Model Runner предъявляет различные требования в зависимости от платформы:

Платформа	Требования
macOS	Требуется Apple Silicon (M1/M2/M3/M4)
Windows	NVIDIA GPU (драйвер 576.57+) или Qualcomm Adreno
Linux	CPU, NVIDIA (CUDA), AMD (ROCm) или Vulkan

Рекомендации по объёму памяти

Размер модели	Требуемый объём RAM	Оптимально для
1–3B параметров	8 ГБ	Быстрые ответы, простые задачи
7–8B параметров	16 ГБ	Универсальный помощник, помощь с кодом
13B+ параметров	32 ГБ+	Сложные рассуждения, длинный контекст

Совет: Если вы не уверены, начните с модели на 7B, например `ai/qwen2.5-coder` — она обеспечивает хороший баланс между качеством и потреблением ресурсов.

Пошаговая настройка

1) Установите или обновите Docker Desktop

Docker Model Runner требует Docker Desktop версии 4.40 или новее.

macOS:

```
# Check current version
docker --version

# Update via brew (if installed via brew)
brew upgrade --cask docker

# Or download from https://www.docker.com/products/docker-desktop/
```

Windows/Linux: Загрузите с https://www.docker.com/products/docker-desktop/

2) Включите Docker Model Runner

1. Откройте Docker Desktop
2. Перейдите в **Settings** (значок шестерёнки)
3. Перейдите в **Features in development ⊠ Beta features**
4. Включите **Docker Model Runner**
5. Нажмите **Apply & restart**

Альтернативный способ (CLI):

```
docker desktop enable model-runner --tcp 12434
```

3) Убедитесь, что Docker Model Runner работает

```
# List available models
docker model list

# Should show available models (empty initially)
```

4) Загрузите модель

Выберите модель, подходящую для вашего оборудования и сценария использования:

```
# Recommended: GLM 4.7 Flash (fast coding assistance)
docker model pull glm-4.7-flash

# Alternative: Qwen3 Coder (agentic coding workflows)
docker model pull qwen3-coder

# Alternative: GPT-OSS (complex reasoning & scheduling)
docker model pull gpt-oss
```

Другие популярные модели:

```
# Qwen 2.5 Coder (proven coding model)
docker model pull ai/qwen2.5-coder

# Llama 3.2 (general purpose)
docker model pull ai/llama3.2

# Mistral (fast, efficient)
docker model pull ai/mistral
```

5) Протестируйте модель напрямую

```
# Run a quick test
docker model run glm-4.7-flash "What is a recursive function?"
```

Вы должны увидеть ответ, сгенерированный локально.

6) Настройте OpenClaw для использования Docker Model Runner

```
# Set provider to OpenAI-compatible mode
openclaw config set provider.name openai

# Point to Docker Model Runner's API
openclaw config set provider.baseUrl http://model-runner.docker.internal/v1

# Set the model name (must match what you pulled)
openclaw config set provider.model glm-4.7-flash

# API key is not needed but the field must exist
openclaw config set provider.apiKey "not-needed"
```

7) Проверьте конфигурацию OpenClaw

```
# Check status
openclaw status

# Test a message
openclaw message send --to self "Hello, are you running locally?"
```

8) Запустите аудит безопасности

```
openclaw security audit --deep
```

Если обнаружены проблемы:

```
openclaw security audit --fix
```

Рекомендуемые модели для OpenClaw

Модели, рекомендуемые Docker (2025)

На основе официальной публикации Docker о приватном ИИ с Clawdbot + DMR:

Модель	Оптимально для	Команда загрузки
`glm-4.7-flash`	Быстрая помощь с кодом и отладка	`docker model pull glm-4.7-flash`
`qwen3-coder`	Агентные рабочие процессы разработки	`docker model pull qwen3-coder`
`gpt-oss`	Сложные рассуждения и планирование	`docker model pull gpt-oss`

Другие популярные модели

Модель	Размер	Оптимально для	Команда загрузки
`ai/qwen2.5-coder`	7B	Разработка, технические задачи	`docker model pull ai/qwen2.5-coder`
`ai/llama3.2`	3B/8B	Общение общего назначения	`docker model pull ai/llama3.2`
`ai/mistral`	7B	Быстрые ответы, сбалансированная модель	`docker model pull ai/mistral`
`ai/gemma2`	9B	Открытая модель Google	`docker model pull ai/gemma2`
`ai/phi-4`	14B	Модель Microsoft для рассуждений	`docker model pull ai/phi-4`
`ai/deepseek-r1`	Различные	Продвинутые рассуждения (крупная модель)	`docker model pull ai/deepseek-r1`

Для большинства пользователей: Начните с `glm-4.7-flash` для быстрой помощи с кодом или `qwen3-coder` для агентных рабочих процессов — это текущие рекомендации Docker, хорошо подходящие для сценариев использования OpenClaw в качестве технического помощника.

Параметры конфигурации

Настройка размера контекста

Некоторые модели поддерживают настраиваемые контекстные окна. Это задаётся через конфигурацию модели:

```
# Check model details
docker model inspect glm-4.7-flash
```

GPU-ускорение

DMR автоматически использует доступное GPU-ускорение:

- **macOS:** Metal (Apple Silicon)
- **Windows/Linux:** CUDA (NVIDIA) или ROCm (AMD)

Дополнительная конфигурация не требуется — Docker Desktop автоматически обнаруживает и использует ваш GPU.

Ограничения ресурсов

Docker Desktop позволяет настраивать ограничения ресурсов для Model Runner:

1. Откройте настройки Docker Desktop
2. Перейдите в раздел **Resources**
3. Настройте ограничения **Memory** и **CPU** по необходимости

Рекомендация: Выделите не менее 8 ГБ RAM для моделей 7B, 16 ГБ для моделей 13B+.

Комбинирование с облачным резервным вариантом

Вы можете настроить OpenClaw на использование локальных моделей для большинства запросов с переключением на облачных провайдеров для сложных задач.

Вариант 1: Ручное переключение

```
# Switch to local
openclaw config set provider.baseUrl http://model-runner.docker.internal/v1
openclaw config set provider.model glm-4.7-flash

# Switch to cloud
openclaw config set provider.baseUrl https://api.anthropic.com
openclaw config set provider.model claude-sonnet-4-20250514
```

Вариант 2: Использование профилей

```
# Create a local profile
OPENCLAW_PROFILE=local openclaw config set provider.baseUrl
 ↪  http://model-runner.docker.internal/v1

# Create a cloud profile
OPENCLAW_PROFILE=cloud openclaw config set provider.baseUrl
 ↪  https://api.anthropic.com

# Run with specific profile
OPENCLAW_PROFILE=local openclaw gateway run
```

Ограничения и компромиссы

Качество моделей

Локальные модели (7B–13B параметров) в целом менее производительны, чем передовые облачные модели (Claude Opus, GPT-4o):

Возможность	Локальная 7B	Передовая облачная
Простые вопросы и ответы	Хорошо	Отлично
Генерация кода	Хорошо	Отлично
Сложные рассуждения	Ограниченно	Отлично
Длинный контекст	Ограниченно (обычно 4K–8K)	Большой (100K+)
Использование инструментов/вызов функций	Зависит от модели	Отлично

Потребление ресурсов

- **RAM:** Модели загружаются в память; ожидайте использование 8–16 ГБ+
- **Диск:** Каждая модель занимает 2–20 ГБ на диске
- **GPU VRAM:** При использовании GPU-ускорения модель размещается в VRAM
- **Первый ответ:** Холодный старт занимает 10–30 секунд, пока модель загружается

Надёжность

- Локальные модели могут иногда выдавать ответы более низкого качества или непоследовательные ответы
- Автоматический повтор/переключение на резерв без дополнительной настройки отсутствует
- Обновление моделей требует ручной загрузки

Устранение неполадок

«Connection refused» или «Cannot connect to model-runner»

1. Убедитесь, что Docker Desktop запущен
2. Убедитесь, что Model Runner включён в настройках Docker Desktop
3. Проверьте, что модель загружена: `docker model list`
4. Проверьте доступность API:

   ```
   curl http://localhost:12434/v1/models
   ```

Медленные ответы

- Проверьте доступный объём RAM (модель может использовать подкачку на диск)
- Попробуйте модель меньшего размера / более быструю модель (`glm-4.7-flash` оптимизирована для скорости)
- Убедитесь, что GPU-ускорение работает:

  ```
  docker model inspect glm-4.7-flash
  # Look for "accelerator" field
  ```

Модель не найдена

Убедитесь, что имя модели в конфигурации OpenClaw точно совпадает с тем, что вы загрузили:

```
# List pulled models
docker model list

# Verify config
openclaw config get provider.model
```

Высокое потребление памяти

Модели остаются загруженными для быстрых ответов. Чтобы выгрузить:

```
# Stop all running models
docker model stop --all
```

Или настройте таймаут автоматической выгрузки в настройках Docker Desktop.

OpenClaw не может подключиться к API

Специальное имя хоста `model-runner.docker.internal` работает только изнутри Docker. Если OpenClaw запущен вне Docker:

```
# Use localhost with the exposed port instead
openclaw config set provider.baseUrl http://localhost:12434/v1
```

Вопросы конфиденциальности

Что обеспечивает DMR

- **Отсутствие передачи в облако:** Инференс выполняется полностью на вашей машине
- **Возможность отключения телеметрии:** Docker Desktop может быть настроен для отключения телеметрии
- **Отсутствие раскрытия API-ключей:** Для локального инференса не нужно управлять секретами

За что вы по-прежнему несёте ответственность

- **Транскрипты сессий:** По-прежнему хранятся локально в `~/.openclaw/`
- **Токены каналов:** Токены WhatsApp/Telegram по-прежнему необходимы
- **Сетевая доступность:** Держите Gateway только на loopback-интерфейсе
- **Шифрование диска:** Включите FileVault/LUKS для защиты учётных данных

Конфигурация максимальной конфиденциальности

Для наиболее конфиденциальной конфигурации:

```
# Run Gateway loopback-only
openclaw config set gateway.bind loopback

# Use local model
openclaw config set provider.baseUrl http://model-runner.docker.internal/v1
openclaw config set provider.model glm-4.7-flash

# Enable Docker sandbox for tool execution
openclaw config set agents.defaults.sandbox docker
openclaw config set agents.defaults.sandboxNetwork none
```

В сочетании с шифрованием диска FileVault/LUKS это обеспечивает выполнение всей обработки ИИ и хранение данных на оборудовании, которое вы контролируете.

Чек-лист безопасности (Docker Model Runner)

Docker Desktop

- ☐ Установлен Docker Desktop версии 4.40+
- ☐ Функция Model Runner включена
- ☐ Ограничения ресурсов настроены надлежащим образом
- ☐ Телеметрия Docker Desktop отключена (при необходимости)

Конфигурация OpenClaw

- ☐ `provider.baseUrl` указывает на Model Runner
- ☐ `provider.model` совпадает с загруженной моделью
- ☐ `gateway.bind` установлен в `loopback`
- ☐ Политика DM установлена в `allowlist` или `pairing`

Управление моделями

- ☐ Модель загружена и работоспособность проверена
- ☐ Модель соответствует доступному объёму RAM
- ☐ GPU-ускорение проверено (при наличии)

Общая безопасность

- ☐ Шифрование диска FileVault/LUKS включено
- ☐ Права доступа к `~/.openclaw/` установлены в 0700
- ☐ Защита истории командной оболочки включена
- ☐ Аудит `openclaw security audit --deep` пройден

Сравнение затрат

Развёртывание	Стоимость настройки	Ежемесячные затраты	Примечания
DMR	Оборудование (имеющееся)	$0	Только пропускная способность для загрузки модели
Облако (Anthropic)	$0	$5–50+	Оплата за токен
VPS + Облако	$0	$6–20+	VPS + затраты на API
Moltworker	$0	$5–15	Тарифный план Cloudflare Workers

Итог: Если у вас уже имеется производительное оборудование (Mac на Apple Silicon, игровой ПК с GPU), DMR обеспечивает бессрочный локальный ИИ при нулевых предельных затратах.

Дальнейшие шаги

После настройки:

1. **Протестируйте выполнение инструментов** — убедитесь, что инструменты web fetch и exec работают корректно
2. **Экспериментируйте с моделями** — попробуйте различные модели для ваших сценариев использования
3. **Настройте каналы** — настройте WhatsApp/Telegram как обычно
4. **Проверьте транскрипты** — убедитесь, что качество ответов соответствует вашим потребностям

Настройка каналов описана в: Pairing Guide

Оптимизация локальных моделей: https://docs.openclaw.ai/gateway/local-models

Модель угроз (для начинающих)

Эта страница объясняет, *почему* конфигурация приватности и безопасности важна для OpenClaw.

Если вы возьмёте из этого только одну мысль:

> Считайте **хост Gateway** (и всё, до чего он может добраться) границей безопасности.

Почему OpenClaw «отличается по рискам» от обычного чат-бота

OpenClaw может быть настроен на гораздо большее, чем просто текстовые ответы.

В зависимости от того, что вы включите, он может:

- отправлять исходящие сообщения через реальные аккаунты мессенджеров
- запрашивать или просматривать ненадёжные веб-сайты
- запускать запланированные задания (автоматизация)
- вызывать инструменты, работающие с файлами/сетью
- вызывать действия узлов/устройств (камера/аудио/canvas)

Это означает, что атакующему не нужно «взламывать» кодовую базу для причинения вреда. Зачастую достаточно:

1) отправить сообщение (или веб-страницу) в контекст агента, и
2) убедить модель совершить небезопасное действие

Это практическая форма **инъекции промптов**.

Официальная документация по безопасности (первоисточник): https://docs.openclaw.ai/gateway/securit

---

Поверхности угроз

1) Входящие сообщения (ЛС и группы)

- незнакомцы, пишущие боту в ЛС
- ненадёжные участники группового канала
- «приманка упоминанием» (тегирование бота с инструкциями)

Меры защиты:

- привязка ЛС / списки разрешённых
- списки разрешённых групп + требование упоминания по умолчанию
- ограничение команд / группы доступа

2) Ненадёжный контент, который бот читает

Даже если только вы можете писать боту, бот может читать ненадёжный контент:

- веб-страницы (fetch/search/browser)
- вложения (PDF, документы)
- вставленные логи
- пересланные сообщения

Меры защиты:

- относитесь к ненадёжному контенту так же, как к вложению электронной почты
- используйте «читающего» агента с отключёнными инструментами
- избегайте сочетания «читает случайные веб-страницы» с «может запускать exec»

3) Сетевая экспозиция

Если Gateway доступен из большего количества сетей, чем планировалось, риск резко возрастает.

Меры защиты:

- где возможно, используйте `gateway.bind: "loopback"`
- используйте SSH-туннель / Tailscale Serve
- требуйте аутентификацию (токен/пароль) для привязок, отличных от loopback
- будьте осторожны с обратными прокси; настройте доверенные прокси

Доверенные прокси (конфигурация обратного прокси)

Если вы размещаете обратный прокси (например, nginx, Caddy, Traefik) перед Gateway, две вещи могут пойти не так:

1. **Подделка локального доступа:** Без конфигурации доверенных прокси атакующий может отправить фальшивый заголовок `X-Forwarded-For: 127.0.0.1`, чтобы обмануть Gateway и заставить его считать запрос локальным (обход аутентификации).
2. **IP-проверки видят IP прокси:** Gateway видит IP прокси вместо реального IP клиента, что нарушает ограничение скорости и контроль доступа.

Gateway решает это через цепочку доверия:

- `isTrustedProxyAddress()` проверяет, находится ли подключающийся IP в вашем списке доверенных (`src/gateway/net.ts:142-148`)
- `resolveGatewayClientIp()` читает заголовки `X-Forwarded-For`/`X-Real-IP` только когда непосредственное соединение приходит от доверенного прокси (`src/gateway/net.ts:150-164`)
- `isLocalDirectRequest()` использует обе проверки для определения, действительно ли запрос локальный (`src/gateway/auth.ts:96-117`)

Настройка:

```
openclaw config set gateway.trustedProxies '["127.0.0.1"]'
```

Практические правила:

- Используете обратный прокси? Установите `trustedProxies` на IP-адрес(а) прокси
- Нет обратного прокси? Оставьте `trustedProxies` пустым (по умолчанию)
- `openclaw security audit` отмечает это как `gateway.trusted_proxies_missing`, когда прокси обнаружен, но не настроен

Исходный код: `src/gateway/auth.ts:73-94` (`resolveTailscaleClientIp()`, `resolveRequestClientIp()`), `src/security/audit.ts` (проверка аудита)

4) Локальный диск и секреты

OpenClaw хранит транскрипты и учётные данные на диске в каталоге `~/.openclaw/`. Если другой пользователь/процесс на хосте может читать этот каталог, приватность утрачена.

Меры защиты:

- права доступа к файлам (аудит исправляет их)
- избегайте синхронизации `~/.openclaw` в облачные хранилища
- защита на уровне ОС (отдельный пользователь, шифрование диска)

5) Цепочка поставок / плагины

Плагины выполняются в контексте процесса, а установка плагина может запускать код во время установки.

Меры защиты:

- устанавливайте только доверенные плагины
- фиксируйте версии
- проверяйте код на диске
- ведите список разрешённых (если поддерживается)

Маркетплейс ClawHub

ClawHub — сторонний маркетплейс навыков для OpenClaw. В феврале 2026 было обнаружено **341 вредоносный навык** (12% от проверенных пакетов). Кампания «ClawHavoc» использовала социальную инженерию (поддельные команды предварительных требований) для установки вредоноса Atomic Stealer.

Ключевой риск: Навыки выполняются внутри процесса Gateway. Вредоносный навык имеет полный доступ к учётным данным, сессиям и сети.

Улучшения безопасности (февраль 2026): ClawHub теперь сканирует все опубликованные навыки через партнёрство с VirusTotal (6-этапный конвейер с 70+ антивирусными движками + ревью кода Gemini LLM, ежедневные повторные сканирования). OpenClaw также включает встроенный локальный сканер навыков (`src/security/skill-scanner.ts`), который выполняет статический анализ на основе паттернов при установке, обнаруживая опасные паттерны: выполнение оболочки, eval, крипто-майнинг и сбор учётных данных.

Оговорка: Ни один из сканеров не может обнаружить социальную инженерию (вектор атаки ClawHavoc использовал поддельные команды «предварительных требований») или полезные нагрузки инъекции промптов. Чистое сканирование не является гарантией безопасности.

Дополнительные меры защиты для ClawHub:

- Проверяйте статус сканирования VirusTotal на странице навыка ClawHub перед установкой
- Обращайте внимание на предупреждения локального сканера при установке навыков
- Избегайте навыков младше 30 дней
- Никогда не запускайте «предварительные» команды терминала из документации навыков
- Используйте Koi Security Scanner как независимую стороннюю проверку
- Проверяйте код навыков в `~/.openclaw/skills/` перед включением

6) mDNS/Bonjour обнаружение в сети

Gateway может анонсировать себя в локальной сети через mDNS (Bonjour), используя тип сервиса `_openclaw-gw._tcp`. Это упрощает автоматическое обнаружение Gateway клиентами OpenClaw (например, приложение для Mac, мобильные приложения), но также означает, что любой в той же LAN может видеть широковещательное объявление.

Что транслируется (поля TXT-записи):

Поле	Всегда включено	Только в full-режиме
`gatewayPort`	Да	Да
`transport` (“gateway”)	Да	Да
`canvasPort`	Да (если задан)	Да (если задан)
`tailnetDns`	Да (если задан)	Да (если задан)
`gatewayTlsSha256`	Да (если TLS)	Да (если TLS)
`cliPath`	–	Да
`sshPort`	–	Да

Исходный код: `src/infra/bonjour.ts:12-26` (тип opts), `src/infra/bonjour.ts:130-146` (минимальные условия), `src/gateway/server-discovery-runtime.ts:19,23-30` (логика mdnsMode)

Риск: В режиме «full» широковещание включает `cliPath` (структуру файловой системы) и `sshPort` (вектор атаки). Даже в режиме «minimal» порт Gateway и имя хоста видны любому в сегменте LAN.

Меры защиты:

- Режим по умолчанию — `minimal` (без cliPath и sshPort)
- Полное отключение: `openclaw config set discovery.mdns off`
- Переменная окружения: `OPENCLAW_DISABLE_BONJOUR=1`
- mDNS работает только в пределах канального уровня (не маршрутизируется за пределы LAN/подсети)
- Используйте режим «full» только в доверенных домашних сетях для отладки

7) Файлы персистентной памяти

OpenClaw загружает девять именованных файлов рабочего пространства `.md` (AGENTS.md, SOUL.md, TOOLS.md, IDENTITY.md, USER.md, HEARTBEAT.md, BOOTSTRAP.md, MEMORY.md, memory.md) при каждом ходе агента через `loadWorkspaceBootstrapFiles()` (`src/agents/workspace.ts:276-330`). Они внедряются напрямую в системный промпт как доверенный контекст — они **не** содержат маркеров `<<<EXTERNAL_UNTRUSTED_CONTENT>>>` которыми помечаются запрошенные веб-страницы или полезные нагрузки вебхуков. Каждый файл усекается до 20 000 символов (`src/agents/pi-embedded-helpers/bootstrap.ts:84`

Дополнительно, файлы каталога `memory/*.md` доступны через вызовы инструментов `memory_search`/`memory_get` (бюджет 4 000 символов) через отдельный конвейер (`src/memory/internal.ts:78-107`).

Риск: Любой процесс или пользователь с доступом на запись в рабочее пространство может внедрить персистентную инъекцию промптов, которая сохраняется между сессиями и выглядит как доверенные системные инструкции. Встроенный сканер навыков

(`src/security/skill-scanner.ts:37-46`) проверяет только файлы JS/TS — ни один из этих `.md` файлов не сканируется на вредоносный контент.

Меры защиты:

- Права доступа на уровне ОС (ограничьте запись в каталог рабочего пространства)
- Периодический аудит содержимого: `grep -rn "<!--" .` для обнаружения скрытых HTML-комментариев
- Запускайте сканер Cisco AI Defense для более глубокого анализа каталога рабочего пространства на основе LLM
- Экспозиция субагентов ограничена: `filterBootstrapFilesForSession()` (`src/agents/workspace.ts:334-342`) предоставляет субагентам доступ только к AGENTS.md + TOOLS.md

Исходный код: `src/agents/workspace.ts:30-31` (список файлов), `src/agents/pi-embedded-hel` (внедрение), `src/memory/qmd-manager.ts:346-352` (валидация QMD)

Практическая модель атакующего

Спросите: «кто реально может причинить проблемы?»

- Случайные пользователи интернета (если вы открыли ЛС/группы)
- Участники ваших групповых чатов
- Скомпрометированный плагин/пакет
- Скомпрометированный аккаунт провайдера модели / API-ключ
- Скомпрометированный хост (взлом VPS, украденный ноутбук)

Основной принцип: контроль доступа важнее интеллекта

Документация OpenClaw подчёркивает порядок, который работает на практике:

1) **Сначала идентичность** — кому разрешено активировать бота (привязка/списки разрешённых)
2) **Затем область действия** — что боту разрешено делать (политика инструментов/песочница/узлы)
3) **Модель в последнюю очередь** — предполагайте, что моделью можно манипулировать; проектируйте так, чтобы манипуляция имела ограниченный радиус поражения

Ограничение доступа по агентам (многоагентные конфигурации)

При запуске нескольких агентов на одном Gateway каждый агент должен иметь только необходимый ему доступ. OpenClaw предоставляет три уровня ограничения:

Изоляция песочницы для каждого агента

Каждый агент может иметь собственную конфигурацию песочницы, контролирующую:

- **mode:** `off` (без песочницы), `agent` (контейнер на агента) или `all` (все сессии в песочнице)
- **scope:** `dedicated` (изолированный) или `shared` (общий с другими агентами)
- **workspaceAccess:** `none`, `ro` (только чтение) или `rw` (чтение-запись)
- **Docker:** запускается ли песочница в Docker-контейнере

Настройки конкретного агента переопределяют глобальные значения по умолчанию. Порядок разрешения: конфигурация агента -> глобальные значения по умолчанию -> встроенные значения по умолчанию.

Исходный код: `src/agents/sandbox/config.ts:126` (`resolveSandboxConfigForAgent()`)

Политики инструментов для каждого агента

Каждый агент может иметь собственный список разрешённых/запрещённых инструментов. Цепочка приоритетов: 1. Разрешённые/запрещённые для конкретного агента (`agents.list[].tools.sandbox.tools`) 2. Глобальные значения по умолчанию (`tools.sandbox.tools`) 3. Встроенные значения по умолчанию (все инструменты разрешены)

Если списки конкретного агента заданы, они имеют приоритет над глобальными.

Исходный код: `src/agents/sandbox/tool-policy.ts:71` (`resolveSandboxToolPolicyForAgent`

Именованные профили инструментов

Для удобства OpenClaw предоставляет именованные профили инструментов для типовых сценариев:

Профиль	Описание	Что разрешено
`minimal`	Только чтение статуса	Только `session_status`
`coding`	Задачи разработки	Файловая система, среда выполнения, сессии, память, изображения
`messaging`	Задачи коммуникации	Группа обмена сообщениями, управление сессиями
`full`	Без ограничений	Все инструменты (пустой список разрешённых = без ограничений)

Исходный код: `src/agents/tool-policy.ts:63-80` (`TOOL_PROFILES`)

Изоляция сессий ЛС

Сессии ЛС каждого агента привязаны к ключу `<agentId>:<sessionKey>`, поэтому диалоги с агентом A не могут просочиться в контекст агента B — даже если оба агента обслуживают один и тот же канал обмена сообщениями.

Рекомендации по безопасности для многоагентных конфигураций

- Применяйте **принцип минимальных привилегий**: каждый агент получает только те инструменты и доступ к рабочему пространству, которые ему нужны
- Используйте `sandbox.mode: "all"` для агентов, обрабатывающих ненадёжный ввод (например, публичные боты)
- Устанавливайте `workspaceAccess: "none"` или `"ro"` для агентов, которым не нужно записывать файлы
- Используйте профили инструментов `minimal` или `messaging` для агентов, которым нужны только коммуникационные возможности

Что означает «высокая приватность» в терминах OpenClaw

Высокая приватность обычно подразумевает:

- хост Gateway защищён
- Gateway привязан к localhost (или только к tailnet)
- панель управления не является публичной
- только вы (или небольшой список разрешённых) можете активировать бота
- поверхность инструментов минимальна
- чувствительные логи отредактированы
- учётные данные хранятся в переменных окружения или зашифрованных хранилищах, где это возможно

Чек-лист усиления безопасности (высокая приватность)

Это практический чек-лист для достижения надёжной позиции «высокая приватность / минимальная экспозиция».

Первоисточники:

- https://docs.openclaw.ai/gateway/security
- https://docs.openclaw.ai/gateway/remote
- https://docs.openclaw.ai/gateway/tailscale
- https://docs.openclaw.ai/start/pairing

---

0) Определите границу доверия (рекомендуется)

- Считайте **хост Gateway** границей доверия.
- Если вы не можете укрепить хост, не включайте мощные инструменты.

---

1) Ограничьте, кто может активировать бота (ЛС)

Предпочтительная политика ЛС: pairing (или allowlist)

- `pairing`: неизвестные отправители получают код; бот игнорирует их до одобрения
- `allowlist`: неизвестные отправители блокируются
- избегайте `open`

Одобрение запросов привязки:

```
openclaw pairing list telegram
openclaw pairing approve telegram <CODE>
```

Документация: https://docs.openclaw.ai/start/pairing

---

2) Ограничьте поведение в группах

Типичные безопасные настройки для групп:

- требовать упоминание
- ограничить, в каких группах бот будет отвечать
- ограничить, кто может вызывать команды в группах

Документация:

- https://docs.openclaw.ai/concepts/groups
- https://docs.openclaw.ai/gateway/security

---

3) Держите Gateway только на loopback, если нет причин иначе

Рекомендуемое значение по умолчанию:

- `gateway.bind: "loopback"`

Схемы удалённого доступа:

SSH-туннель (универсальный)

```
ssh -N -L 18789:127.0.0.1:18789 user@gateway-host
```

Tailscale Serve (лучший UX; только в tailnet)

Оставьте Gateway на loopback и откройте UI через HTTPS Serve.

Документация: https://docs.openclaw.ai/gateway/tailscale

---

4) Требуйте аутентификацию для любого не-loopback доступа

- Аутентификация по токену/паролю обязательна при привязке за пределами loopback.
- Мастер настройки генерирует токен по умолчанию.

Если вы подозреваете ошибку в конфигурации аутентификации:

- используйте `openclaw dashboard` для получения URL с одноразовым токеном
- запустите `openclaw security audit` для обнаружения рискованной экспозиции

---

5) Регулярно запускайте аудит безопасности

```
openclaw security audit
openclaw security audit --deep
openclaw security audit --fix
```

`--fix` устраняет типичные ошибки (групповая политика, редактирование, права на файлы).

Документация: https://docs.openclaw.ai/gateway/security

---

6) Относитесь к управлению браузером как к административному API

Если вы включаете удалённое управление браузером:

- предпочтительно только через tailnet
- требуется аутентификация по токену
- избегайте Funnel, если вы *явно* не хотите публичной экспозиции

Документация: https://docs.openclaw.ai/gateway/security и https://docs.openclaw.ai/gateway/tailscale

---

7) Минимизируйте радиус поражения инструментов

Практические рекомендации:

- Начните с отключённых или минимальных инструментов.
- Добавляйте одну категорию инструментов за раз.
- Предпочитайте песочницу для не-основных сессий.

Документация:

- https://docs.openclaw.ai/tools
- https://docs.openclaw.ai/gateway/sandboxing

  Почему это важно для инъекций промптов: Ограничение инструментов сокращает ущерб, который может нанести успешная инъекция.

---

8) Защитите секреты на диске

- Относитесь к `~/.openclaw` как к конфиденциальному каталогу.
- Не синхронизируйте его в iCloud/Dropbox и т.д.
- Убедитесь, что права доступа строгие (аудит может исправить).

Документация: https://docs.openclaw.ai/gateway/security

---

9) Будьте осторожны с плагинами/расширениями

Плагины выполняются **внутри процесса**.

Рекомендации:

- устанавливайте только доверенные плагины
- фиксируйте версии
- проверяйте код на диске

Предупреждение о навыках ClawHub

В феврале 2026 на ClawHub было обнаружено **341 вредоносный навык** (12% от проверенных пакетов). Атака использовала социальную инженерию, а не эксплойты кода.

Улучшения сканирования (февраль 2026): ClawHub теперь сканирует все опубликованные навыки через партнёрство с VirusTotal (автоматический анализ + ежедневные повторные сканирования). OpenClaw также включает встроенный локальный сканер навыков, запускающийся при установке и обнаруживающий опасные паттерны кода. Однако ни один сканер не может отловить социальную инженерию (реальный вектор атаки ClawHavoc) или инъекцию промптов — ручная проверка остаётся необходимой.

Перед установкой любого навыка ClawHub:

- ☐ Проверьте статус сканирования VirusTotal на странице навыка
- ☐ Просмотрите предупреждения локального сканера при установке
- ☐ Проверьте возраст навыка (избегайте младше 30 дней)
- ☐ Убедитесь в репутации издателя
- ☐ Прочитайте сам код, а не только документацию
- ☐ **НИКОГДА** не выполняйте «предварительные» команды терминала из документации навыков
- ☐ Используйте Koi Security Scanner как независимую стороннюю проверку
- ☐ Будьте особенно подозрительны к навыкам, связанным с криптовалютой

Документация: https://docs.openclaw.ai/plugin и https://docs.openclaw.ai/gateway/security

---

10) Контролируйте обнаружение mDNS/Bonjour

Gateway может объявлять своё присутствие в локальной сети через mDNS. Проверьте текущий режим:

```
openclaw config get discovery.mdns
```

Сетевое окружение	Рекомендуемый режим	Почему
Домашняя LAN (доверенная)	`minimal` (по умолчанию)	Удобное обнаружение; конфиденциальные поля опущены
Общая/публичная сеть	`off`	Не объявлять gateway вообще
Отладка в доверенной сети	`full`	Открывает cliPath + sshPort для диагностики

Полное отключение mDNS:

```
openclaw config set discovery.mdns off
```

Или через переменную окружения:

```
export OPENCLAW_DISABLE_BONJOUR=1
```

Исходный код: `src/gateway/server-discovery-runtime.ts:19,23-30`, `src/infra/bonjour.ts`

---

11) Настройте доверенные прокси

Если вы используете обратный прокси (nginx, Caddy, Traefik) перед Gateway, вы **должны** настроить доверенные прокси, чтобы Gateway мог корректно определять IP клиентов и применять проверки локального доступа.

```
openclaw config set gateway.trustedProxies '["127.0.0.1"]'
```

Проверка конфигурации:

```
openclaw security audit
# Ищите: gateway.trusted_proxies_missing
```

Если вы не используете обратный прокси, оставьте `trustedProxies` пустым (по умолчанию).

Исходный код: `src/gateway/net.ts:142-164`

---

12) Проверяйте файлы рабочего пространства .md на скрытый контент

OpenClaw загружает девять файлов `.md` рабочего пространства напрямую в системный промпт агента при каждом ходе. Они выглядят как доверенный контекст — не обёрнуты маркерами ненадёжного контента. Встроенный сканер навыков не проверяет файлы `.md`, поэтому вредоносный контент в этих файлах невидим для автоматических проверок.

Сканирование на скрытые HTML-комментарии (наиболее распространённый вектор инъекции в файлах `.md`):

```
grep -rn "<!--" ~/your-workspace-dir/
```

Сканирование на подозрительные паттерны инструкций:

```
grep -rniE "(ignore previous|system override|you are now|execute the
↪  following|curl.*base64|wget.*credentials)" ~/your-workspace-dir/*.md
```

Запуск сканера Cisco AI Defense для глубокого анализа на основе LLM:

```
skill-scanner scan ~/your-workspace-dir/
```

Файлы для аудита: `AGENTS.md`, `SOUL.md`, `TOOLS.md`, `IDENTITY.md`, `USER.md`, `HEARTBEAT.md`, `BOOTSTRAP.md`, `MEMORY.md`, `memory.md` и любые файлы в каталоге `memory/`.

---

13) Никогда не позволяйте ИИ изменять критичные для безопасности настройки

Действия `config.apply` и `config.patch` инструмента gateway не имеют **никаких проверок разрешений** — в отличие от чат-команды `/config set` (которая проверяет и `commands.config`, и `configWrites`), инструмент gateway полностью обходит оба ограничения. Это основной риск самопроизвольного изменения конфигурации ИИ.

Удалите инструмент `gateway` (основная защита):

```
# Вариант A: используйте профиль инструментов coding (исключает gateway
↪  полностью)
openclaw config set tools.profile coding

# Вариант B: запретите инструмент gateway явно
openclaw config set tools.deny '["gateway"]'
```

Исходный код: `src/agents/tool-policy.ts:63-80` (профиль coding), `src/agents/tools/gatewa` (отсутствие проверок)

Оставьте команды конфигурации отключёнными (эшелонированная защита):

```
# commands.config по умолчанию false — убедитесь, что так и осталось
openclaw config get commands.config
# Должно быть: false (или не задано)
```

Установите `configWrites: false` для всех каналов:

```
openclaw config set configWrites false
```

Примечание: `configWrites: false` блокирует только чат-команду `/config set`. Инструмент gateway **не** блокируется — именно поэтому удаление инструмента gateway (выше) является основной защитой.

После любого изменения конфигурации, инициированного ИИ:

```
# Запустите аудит безопасности
openclaw security audit --deep

# Сравните с резервной копией (OpenClaw хранит 5 ротируемых .bak файлов)
diff ~/.openclaw/openclaw.json.bak ~/.openclaw/openclaw.json

# Восстановите из резервной копии при необходимости
cp ~/.openclaw/openclaw.json.bak ~/.openclaw/openclaw.json
openclaw gateway restart
```

Исходный код: `src/config/io.ts:343-362` (ротация резервных копий)

Рассмотрите версионный контроль вашей конфигурации для отслеживания изменений:

```
cd ~/.openclaw && git init && git add openclaw.json && git commit -m "known
 ↪  good baseline"
# После любого изменения: git diff для просмотра что изменилось
```

Журнал аудита конфигурации (добавлен в sync 10): каждая запись конфигурации теперь логируется в `$STATE_DIR/logs/config-audit.jsonl` с PID, PPID, хешами содержимого, размерами в байтах и флагами аномалий. Просмотр:

```
# Показать недавние записи конфигурации с аномалиями
cat ~/.openclaw/logs/config-audit.jsonl | python3 -c "
import sys,json
for line in sys.stdin:
  r = json.loads(line)
  if r.get('suspicious'):
    print(f\"{r['ts']} pid={r['pid']} {r['suspicious']}\")
"
```

Исходный код: `src/config/io.ts:370-413` (вспомогательные функции аудита), `:958-1062` (построитель записей аудита)

---

14) Операционные паттерны (после развёртывания)

Конфигурация безопасности критична, но **как вы используете OpenClaw день за днём** не менее важно. Реальные пользователи сообщают о следующих операционных подводных камнях:

Паттерн	Риск	Решение
Постоянно работающие агенты	Неожиданные счета за API ($847/мес в одном случае)	Установите оповещения о расходах, ежедневно мониторьте использование
Задачи более 10 шагов	40% вероятность сбоя из-за дрейфа контекста	Разбивайте на части, используйте контрольные точки
Неоднозначность «черновика»	Агент интерпретирует «draft» как «send»	Говорите явно «ПОКАЖИ МНЕ, не отправляй»
Общий профиль браузера	Агент наследует ВСЕ ваши сессии	Используйте выделенный профиль OpenClaw
Длинные сессии	Контекстное окно заполняется, агент забывает контекст	Обновляйте каждые 1–2 дня, используйте файлы памяти
Навыки ClawHub	341 вредоносный навык обнаружен (февраль 2026)	Читайте код, проверяйте издателя, сканируйте
Смешение личного и рабочего	Нарушения политики BYOD	Используйте отдельные экземпляры и рабочие пространства

См. также: Пример конфигурации высокой приватности для полной защищённой конфигурации.

Пример конфигурации высокой приватности

> **Назначение:** Это справочная конфигурация для укреплённого развёртывания OpenClaw с высоким уровнем приватности. Сохраните её как `~/.openclaw/openclaw.json` после проверки и адаптации к вашему окружению.

Эта конфигурация предполагает:

- Gateway привязан только к loopback
- Доступ через SSH-туннель или Tailscale
- Ваш провайдер модели настроен через `openclaw onboard`
- Минимальная поверхность инструментов
- Привязка (pairing) обязательна для доступа через ЛС

> **Примечание:** Настройте учётные данные провайдера модели с помощью `openclaw onboard --install-daemon`. Мастер настройки безопасно обрабатывает хранение API-ключей. Не помещайте API-ключи непосредственно в этот конфигурационный файл.

Конфигурационный файл

Каждая опция ниже проверена по определениям типов конфигурации OpenClaw в `src/config/types.*.ts`.

```
{
  "$schema": "https://openclaw.ai/schemas/2024-11/config.json",
  "version": 1,

  // --- Безопасность Gateway ---
  "gateway": {
    "bind": "loopback",          // Слушать только на 127.0.0.1
    "port": 18789,               // Порт по умолчанию
    "auth": {
      "mode": "token",
```

```
    "token": "GENERATE_WITH_OPENSSL_COMMAND_BELOW"
  },
  "trustedProxies": [],        // Пустой = нет обратного прокси
  "controlUi": {
    "dangerouslyDisableDeviceAuth": false  // Оставить аутентификацию
↪ устройства включённой
  }
},

// --- Конфигурация каналов ---
// Настройте по каналам через openclaw onboard или openclaw config set
// Примеры показаны в виде комментариев:

// Telegram - режим привязки (рекомендуется)
// "channels": [{
//   "type": "telegram",
//   "enabled": true,
//   "dmPolicy": "pairing",
//   "groupPolicy": {
//     "requireMention": true,
//     "allowedGroups": ["your-approved-group-id"]
//   },
//   "configWrites": false
// }]

// WhatsApp - режим привязки
// "channels": [{
//   "type": "whatsapp",
//   "enabled": true,
//   "dmPolicy": "pairing"
// }]

// Discord - режим списка разрешённых
// "channels": [{
//   "type": "discord",
//   "enabled": true,
//   "dmPolicy": "allowlist",
//   "allowFrom": ["your-user-id"],
//   "configWrites": false
// }]

// --- Настройки агента по умолчанию: безопасность инструментов ---
"agents": {
  "defaults": {
    "tools": {
      "profile": "minimal"     // Только чтение статуса, без доступа к
↪ инструментам
    },
    "sandbox": {
      "mode": "all",           // Песочница для всех не-основных сессий
      "workspaceAccess": "none" // Без доступа к файловой системе
↪ рабочего пространства
    }
  }
},

// --- Логирование с редактированием ---
"logging": {
```

```
    "redactSensitive": "tools"   // Редактировать ввод/вывод инструментов в
↪   логах
  },

  // --- Безопасность браузера ---
  "browser": {
    "evaluateEnabled": false     // Отключить выполнение JS в браузере
  },

  // --- Обнаружение ---
  "discovery": {
    "mdns": {
      "mode": "off"              // Не объявлять в локальной сети
    }
  },

  // --- Безопасность плагинов ---
  "plugins": {
    "enabled": false             // Без плагинов, пока явно не понадобятся
  },

  // --- Безопасность команд ---
  "commands": {
    "config": false              // Отключить чат-команду /config set
  }
}
```

Описание ключевых настроек безопасности

Привязка Gateway (`gateway.bind`)

Значение	Описание	Уровень риска
`"loopback"`	Слушать только на 127.0.0.1	Минимальный — без внешнего доступа
`"lan"`	Слушать на всех интерфейсах LAN	Средний — требуется аутентификация
`"tailnet"`	Привязка только к IP Tailscale	Низкий — только в tailnet

Исходный код: `src/config/types.gateway.ts`

Политика ЛС (`dmPolicy`)

Значение	Описание
`"pairing"`	Неизвестные отправители получают код привязки; игнорируются до одобрения
`"allowlist"`	Только перечисленные ID могут писать в ЛС; остальные заблокированы

Значение	Описание
`"open"`	Любой может писать в ЛС (НЕ рекомендуется для ботов с инструментами)

Политика групп (`groupPolicy`)

- `requireMention: true` — бот отвечает только при упоминании
- `allowedGroups: ["id1", "id2"]` — отвечать только в этих группах
- `requireCommandPrefix: "!"` — отвечать только на команды, начинающиеся с !

Профили инструментов (`tools.profile`)

Профиль	Описание
`"minimal"`	Только чтение статуса — без доступа к инструментам
`"coding"`	Файловая система, среда выполнения, сессии, память, изображения
`"messaging"`	Группа обмена сообщениями, управление сессиями
`null` (пустой)	Все инструменты доступны

Исходный код: `src/agents/tool-policy.ts:63-80`

Инструкции по настройке

1) Сгенерируйте безопасный токен gateway

```
# Сгенерировать случайный 64-символьный hex-токен
export GATEWAY_AUTH_TOKEN="$(openssl rand -hex 32)"

# Проверить, что он установлен
echo $GATEWAY_AUTH_TOKEN
```

Замените `GENERATE_WITH_OPENSSL_COMMAND_BELOW` на ваш реальный токен.

2) Настройте провайдера модели

Используйте мастер настройки для безопасной конфигурации вашего провайдера:

```
openclaw onboard --install-daemon
```

Мастер выполняет:

- Выбор провайдера (Anthropic, OpenAI, Zhipu AI и др.)
- Хранение API-ключа (переменная окружения или связка ключей)
- Выбор модели по умолчанию
- Установку фонового сервиса

3) Примените конфигурацию

```
# Скопируйте конфигурацию в каталог OpenClaw
cp high-privacy-config.example.json5 ~/.openclaw/openclaw.json

# Отредактируйте, добавив ваши конкретные значения
nano ~/.openclaw/openclaw.json
```

4) Перезапустите Gateway

```
openclaw gateway restart
```

5) Проверьте настройку

```
# Проверить статус Gateway
openclaw gateway status

# Запустить аудит безопасности
openclaw security audit --deep

# Проверить здоровье
openclaw health
```

Примечания по каналам

Telegram

1. Создайте бота через @BotFather:
   - Отправьте `/newbot`
   - Следуйте инструкциям для указания имени и описания
   - Скопируйте токен бота
2. Добавьте ваш ID пользователя в массив `allowFrom`
   - Узнайте свой Telegram user ID: напишите `@userinfobot` в Telegram
   - Добавьте числовой ID в конфигурацию
3. Для привязки, после получения кода:

   ```
   openclaw pairing approve telegram <CODE>
   ```

WhatsApp

1. Запустите `openclaw channels login`
2. Отсканируйте QR-код телефоном
3. Одобрите привязанные устройства при необходимости

Настройка удалённого доступа

SSH-туннель (универсальный)

```
# С вашей локальной машины
ssh -N -L 18789:127.0.0.1:18789 user@gateway-host
```

Затем откройте: http://127.0.0.1:18789/

Tailscale Serve (рекомендуется)

1. Установите Tailscale на хосте Gateway и на вашей локальной машине
2. На хосте Gateway:

   ```
   sudo tailscale serve --bg --https=443 127.0.0.1:18789
   ```

3. Настройте OpenClaw для принятия идентификации Tailscale:

   ```
   "gateway": {
     "tailscale": { "mode": "serve" },
     "auth": {
       "allowTailscale": true
     }
   }
   ```

4. Доступ с вашей локальной машины: https://..ts.net/

Устранение неполадок

Gateway не запускается

```
# Проверить синтаксис конфигурации
openclaw config validate

# Проверить логи
openclaw logs --follow
```

Проблемы аутентификации после изменения конфигурации

```
# Получить URL с токеном
openclaw dashboard

# Запустить аудит безопасности для обнаружения ошибок конфигурации
openclaw security audit --deep
```

Памятка по безопасности

1. **Никогда не коммитьте** `openclaw.json` в git — он может содержать токены
2. **Никогда не делитесь** скриншотами с вашей конфигурацией или токенами
3. **Ротируйте токены** регулярно при подозрении на утечку
4. **Запускайте аудит безопасности** после любых изменений конфигурации: `openclaw security audit --fix`
5. **Делайте резервные копии** рабочей конфигурации перед изменениями (OpenClaw автоматически хранит 5 ротируемых `.bak` файлов)

Официальная документация

- Безопасность Gateway: https://docs.openclaw.ai/gateway/security
- Конфигурация: https://docs.openclaw.ai/gateway/configuration
- Tailscale: https://docs.openclaw.ai/gateway/tailscale
- Привязка: https://docs.openclaw.ai/start/pairing

Обнаружение запросов OpenClaw (для хостинг-сервисов)

Обзор

OpenClaw устанавливает различные идентифицирующие заголовки в исходящих HTTP-запросах через несколько подсистем. Этот документ каталогизирует все идентифицируемые заголовки, сгруппированные по обнаруживаемости, со ссылками на исходный код.

Для кого это:

- **Хостинг-сервисы / API-провайдеры:** хотите идентифицировать, ограничивать по скорости или блокировать трафик OpenClaw
- **Пользователи OpenClaw:** хотите понять, что ваш экземпляр раскрывает о себе в исходящих запросах

Все пути к файлам указаны относительно корня репозитория. Номера строк проверены по текущей кодовой базе.

---

1. Явно идентифицируемые запросы

Эти запросы содержат заголовки, прямо называющие OpenClaw. Любой сервис, получающий их, может тривиально определить источник.

`User-Agent: "OpenClaw-Gateway/1.0"` — загрузка медиафайлов

Когда OpenClaw скачивает медиафайлы (изображения, аудио, видео), прикреплённые к входящим сообщениям, Gateway устанавливает:

```
User-Agent: OpenClaw-Gateway/1.0
```

Исходный код: `src/media/input-files.ts:149`

```
init: { headers: { "User-Agent": "OpenClaw-Gateway/1.0" } },
```

Кто это видит: Любой веб-сервер, размещающий медиафайлы, которыми пользователи делятся с OpenClaw.

`User-Agent: "openclaw"` — GitHub API (установка signal-cli)

При автоустановке зависимости Signal CLI OpenClaw запрашивает информацию о релизах с GitHub:

```
User-Agent: openclaw
Accept: application/vnd.github+json
```

Исходный код: `src/commands/signal-install.ts:124`

```
headers: {
  "User-Agent": "openclaw",
  Accept: "application/vnd.github+json",
},
```

Кто это видит: Серверы GitHub API (только при установке signal-cli).

`User-Agent: "openclaw"` — Anthropic OAuth API

При проверке использования Anthropic через OAuth:

```
User-Agent: openclaw
```

Исходный код: `src/infra/provider-usage.fetch.claude.ts:119`

```
"User-Agent": "openclaw",
```

Кто это видит: Серверы API Anthropic.

`HTTP-Referer` + `X-Title` — API OpenRouter / Perplexity

OpenClaw идентифицирует себя для OpenRouter и Perplexity через заголовки атрибуции приложения:

```
HTTP-Referer: https://openclaw.ai
X-Title: OpenClaw
```

Исходный код: `src/agents/pi-embedded-runner/extra-params.ts:8-9`

```
const OPENROUTER_APP_HEADERS: Record<string, string> = {
  "HTTP-Referer": "https://openclaw.ai",
  "X-Title": "OpenClaw",
};
```

Для веб-поиска Perplexity заголовок более описательный:

```
HTTP-Referer: https://openclaw.ai
X-Title: OpenClaw Web Search
```

Исходный код: `src/agents/tools/web-search.ts:481-482`

```
"HTTP-Referer": "https://openclaw.ai",
"X-Title": "OpenClaw Web Search",
```

Кто это видит: Серверы API OpenRouter и Perplexity.

`MM-API-Source: "OpenClaw"` — API MiniMax VLM

OpenClaw использует кастомный заголовок при вызове API визуально-языковой модели MiniMax:

```
MM-API-Source: OpenClaw
```

Исходный код: `src/agents/minimax-vlm.ts:76`

```
"MM-API-Source": "OpenClaw",
```

Также используется при проверке использования MiniMax:

Исходный код: `src/infra/provider-usage.fetch.minimax.ts:322`

```
"MM-API-Source": "OpenClaw",
```

Кто это видит: Серверы API MiniMax.

`clientInfo.name: "openclaw-acp-client"` — протокол ACP

Когда OpenClaw подключается к серверу ACP (Agent Communication Protocol), он идентифицирует себя на уровне протокола:

```
{ "name": "openclaw-acp-client", "version": "1.0.0" }
```

Исходный код: `src/acp/client.ts:129`

```
clientInfo: { name: "openclaw-acp-client", version: "1.0.0" },
```

Кто это видит: Любой ACP-совместимый сервер агентов, к которому подключается OpenClaw.

---

2. Браузероподобные запросы (сложно обнаружить)

Инструмент WebFetch — просмотр веб-сайтов

Когда инструмент WebFetch запрашивает веб-страницы, он использует User-Agent в стиле Chrome и типичные для браузера заголовки:

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 14_7_2) AppleWebKit/537.36 (K
Accept: */*
Accept-Language: en-US,en;q=0.9
```

Исходный код: `src/agents/tools/web-fetch.ts:40-41`

```
const DEFAULT_FETCH_USER_AGENT =
  "Mozilla/5.0 (Macintosh; Intel Mac OS X 14_7_2) AppleWebKit/537.36
   ↪  (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36";
```

Исходный код: `src/agents/tools/web-fetch.ts:420-425`

```
headers: {
  Accept: "text/markdown, text/html;q=0.9, */*;q=0.1",
  "User-Agent": params.userAgent,
  "Accept-Language": "en-US,en;q=0.9",
},
```

Сложность обнаружения: Высокая. Запрос выглядит как настоящий браузер Chrome на macOS. Однако существуют тонкие подсказки для отпечатка:

- **Отсутствующие заголовки браузера:** Настоящий Chrome отправляет заголовки `Sec-Fetch-Mode`, `Sec-Fetch-Site`, `Sec-Fetch-Dest`, `Sec-Ch-Ua` и `Cookie`. WebFetch не отправляет ни одного из них.
- **Поведение защиты от SSRF:** Внутренняя защита от SSRF может создавать характерные паттерны перенаправлений или тайминга.
- **Статическая версия:** Жёстко закодированная строка Chrome 122 будет всё больше устаревать со временем, что делает возможным обнаружение через анализ возраста версии.

Настраивается: Да — `tools.web.fetch.userAgent` в конфигурации (`src/config/types.tools.t`

---

3. Косвенные отпечатки (слабые сигналы)

Эти запросы не называют OpenClaw, но имеют характеристики, отличающиеся от реального браузерного трафика.

API Brave Search

```
Accept: application/json
X-Subscription-Token: <api_key>
```

Исходный код: `src/agents/tools/web-search.ts:660-663`

```
headers: {
  Accept: "application/json",
  "X-Subscription-Token": params.apiKey,
},
```

Кастомный User-Agent не установлен. `fetch()` в Node.js использует UA-строку по умолчанию (обычно `node` или `undici`), которая отличается от браузеров, но не идентифицирует OpenClaw конкретно.

Кто это видит: Серверы API Brave Search.

API xAI Grok

```
Content-Type: application/json
Authorization: Bearer <api_key>
```

Исходный код: `src/agents/tools/web-search.ts:527-535`

```
headers: {
  "Content-Type": "application/json",
  Authorization: `Bearer ${params.apiKey}`,
},
```

Кастомный User-Agent или referer не установлен. Применяется тот же `fetch()` Node.js по умолчанию.

Кто это видит: Серверы API xAI.

Общие слабые сигналы для всех запросов Node.js fetch

Когда явный User-Agent не установлен, `fetch()` в Node.js раскрывает:

- Небраузерную строку User-Agent (например, `node` или имя библиотеки `undici`)
- Отсутствие заголовков `Cookie`, `Sec-*` и `Referer`, которые всегда отправляет настоящий браузер
- Отсутствие согласования `Accept-Encoding`, которое браузеры обычно включают

Эти сигналы указывают на «автоматизированный HTTP-клиент», но не на «OpenClaw» конкретно.

---

4. Подделанные идентификации для провайдеров

Это внутренние вызовы проверки использования, где OpenClaw имитирует другие клиенты для запроса API использования провайдеров. Они **не являются пользовательскими запросами инструментов** — срабатывают только при проверке использования/квот аккаунта.

Проверка использования GitHub Copilot

```
User-Agent: GitHubCopilotChat/0.26.7
Editor-Version: vscode/1.96.2
X-Github-Api-Version: 2025-04-01
```

Исходный код: `src/infra/provider-usage.fetch.copilot.ts:24`

```
"User-Agent": "GitHubCopilotChat/0.26.7",
```

Проверка использования OpenAI/Codex

```
User-Agent: CodexBar
```

Исходный код: `src/infra/provider-usage.fetch.codex.ts:30`

```
"User-Agent": "CodexBar",
```

Проверка использования Google/Antigravity

```
User-Agent: antigravity
X-Goog-Api-Client: google-cloud-sdk vscode_cloudshelleditor/0.1
```

Исходный код: `src/infra/provider-usage.fetch.antigravity.ts:205-206`

```
"User-Agent": "antigravity",
"X-Goog-Api-Client": "google-cloud-sdk vscode_cloudshelleditor/0.1",
```

Примечание: Эти подделанные идентификации значимы для обсуждения отпечатков приватности, потому что они показывают, как OpenClaw отправляет заголовки, утверждающие принадлежность к другому ПО. Соответствующие провайдеры могут обнаружить несоответствие, если соотнесут строки User-Agent с ожидаемым поведением клиентов.

---

5. Идентификаторы клиента Gateway (только внутренние)

Gateway использует идентификацию клиентов на основе WebSocket для внутреннего протокола управления. Эти идентификаторы **не отправляются через внешний HTTP** — они существуют только в канале связи Gateway - клиент.

Исходный код: `src/gateway/protocol/client-info.ts:1-14`

```
export const GATEWAY_CLIENT_IDS = {
  WEBCHAT_UI: "webchat-ui",
  CONTROL_UI: "openclaw-control-ui",
  WEBCHAT: "webchat",
  CLI: "cli",
  GATEWAY_CLIENT: "gateway-client",
  MACOS_APP: "openclaw-macos",
  IOS_APP: "openclaw-ios",
  ANDROID_APP: "openclaw-android",
  NODE_HOST: "node-host",
  TEST: "test",
  FINGERPRINT: "fingerprint",
  PROBE: "openclaw-probe",
};
```

Каждый клиент также отправляет метаданные:

Исходный код: `src/gateway/protocol/client-info.ts:34-43`

```ts
export type GatewayClientInfo = {
  id: GatewayClientId;
  displayName?: string;
  version: string;
  platform: string;
  deviceFamily?: string;
  modelIdentifier?: string;
  mode: GatewayClientMode;
  instanceId?: string;
};
```

Кто это видит: Только сам процесс Gateway. Не видно внешним сервисам, если Gateway не открыт в сети и атакующий не перехватывает WebSocket-трафик.

---

6. Для хостинг-сервисов: советы по обнаружению

Правила сопоставления (по убыванию охвата)

1. **User-Agent содержит “OpenClaw” или “openclaw”** — ловит загрузку медиа (`OpenClaw-Gateway/1.0`), вызовы GitHub API и Anthropic OAuth
2. **HTTP-Referer = `https://openclaw.ai`** — ловит все запросы через OpenRouter и Perplexity
3. **X-Title содержит “OpenClaw”** — ловит запросы OpenRouter/Perplexity (избыточно с referer, но более конкретно)
4. **MM-API-Source = “OpenClaw”** — ловит вызовы API MiniMax

Что вы не обнаружите

- **Трафик просмотра WebFetch** — намеренно использует User-Agent в стиле Chrome и стандартные заголовки браузера. Для пометки этого трафика потребуется более глубокий поведенческий анализ (отсутствие заголовков `Sec-*`, устаревшая версия Chrome, отсутствие cookies).
- **Вызовы API Brave Search / xAI Grok** — нет заголовков, специфичных для OpenClaw; обнаруживается только через общий отпечаток fetch Node.js.

---

7. Для пользователей OpenClaw: управление вашим отпечатком

Что можно настроить

Настройка	Путь в конфигурации	Эффект
User-Agent WebFetch	`tools.web.fetch.userAgent`	Переопределяет UA по умолчанию в стиле Chrome для веб-просмотра

Что нельзя настроить (жёстко закодировано)

Заголовок	Где устанавливается	Примечания
`User-Agent: OpenClaw-Gateway/1.0`	`src/media/input-files.ts:149`	Загрузка медиафайлов
`User-Agent: openclaw`	`src/commands/signal-install.ts:124`	Установка Signal CLI
`User-Agent: openclaw`	`src/infra/provider-usage.fetch.claude.ts:119`	Проверка использования Anthropic
`HTTP-Referer: https://openclaw.ai`	`src/agents/pi-embedded-runner/extra-params.ts:78`	OpenRouter/Perplexity
`X-Title: OpenClaw`	`src/agents/pi-embedded-runner/extra-params.ts:79`	OpenRouter/Perplexity
`X-Title: OpenClaw Web Search`	`src/agents/tools/web-search.ts:482`	Поиск Perplexity
`MM-API-Source: OpenClaw`	`src/agents/minimax-vlm.ts:76`	MiniMax VLM

Рекомендации

Если отпечаток запросов важен для вашего случая использования: 1. Проверьте, какие инструменты и провайдеры у вас включены — каждый несёт собственные идентифицирующие заголовки 2. Используйте `tools.web.fetch.userAgent` для установки актуальной, реалистичной строки браузера для веб-просмотра 3. Помните, что заголовки атрибуции OpenRouter/Perplexity жёстко закодированы и всегда будут идентифицировать ваш трафик как OpenClaw 4. Заголовки загрузки медиа жёстко закодированы — любой сервер, размещающий файлы, которыми вы делитесь с OpenClaw, увидит `OpenClaw-Gateway/1.0`

---

8. Примеры правил Cloudflare WAF

Готовые к использованию выражения кастомных правил Cloudflare WAF для обнаружения, проверки или блокировки трафика OpenClaw. Эти правила используют язык правил Cloudflare и работают на всех тарифных планах (если не указано иное).

Ключевые поля выражений

Поле	Тип	Назначение
`http.user_agent`	String	Заголовок запроса User-Agent
`http.referer`	String	Заголовок запроса Referer
`http.request.headers[name]`	Map«String, Array<String»	Доступ к любому заголовку по имени (**должен быть в нижнем регистре**)
`lower()`	Function	Регистронезависимое сопоставление
`any()`	Function	Проверка совпадения любого элемента массива
`len()`	Function	Проверка наличия заголовка (`len(...) > 0`)
`contains`	Operator	Поиск подстроки (**с учётом регистра**)
`matches`	Operator	Regex-сопоставление — **только Business и Enterprise** (движок Rust regex)

Доступность действий по тарифу

Действие	Значение API	Требуемый план	Применение
Block	`block`	Free+	Жёсткая блокировка запросов OpenClaw
Managed Challenge	`managed_challenge`	Free+	Адаптивная проверка (рекомендуемый первый шаг)
JS Challenge	`js_challenge`	Free+	JavaScript-проверка
Interactive Challenge	`challenge`	Free+	Проверка в стиле CAPTCHA
Log	`log`	**Только Enterprise**	Логировать без блокировки — сначала оцените влияние правила
Skip	`skip`	Free+	Исключить известный надёжный трафик OpenClaw из других правил

Правило 1: Обнаружить все явно идентифицируемые запросы OpenClaw (широкое совпадение)

```
(lower(http.user_agent) contains "openclaw") or
(http.referer contains "openclaw.ai") or
(any(http.request.headers["x-title"][*] contains "OpenClaw")) or
(any(http.request.headers["mm-api-source"][*] eq "OpenClaw"))
```

Рекомендуемое действие: `managed_challenge` для начального развёртывания, затем переключение на `block` после проверки событий безопасности.

Правило 2: Совпадение только по идентификаторам User-Agent (простейшее)

```
(lower(http.user_agent) contains "openclaw")
```

Ловит: `OpenClaw-Gateway/1.0`, `openclaw`.

Правило 3: Совпадение по referer-атрибуции OpenRouter/Perplexity

```
(http.referer eq "https://openclaw.ai")
```

Ловит: все запросы, маршрутизированные через OpenRouter или Perplexity с атрибуцией OpenClaw.

Правило 4: Совпадение по идентификаторам кастомных заголовков

```
(len(http.request.headers["x-title"]) > 0 and any(http.request.headers["x-title"
(len(http.request.headers["mm-api-source"]) > 0 and any(http.request.headers["mm
```

Правило 5: Подход Enterprise с логированием (проверка перед блокировкой)

```
(lower(http.user_agent) contains "openclaw") or
(http.referer contains "openclaw.ai") or
(any(http.request.headers["x-title"][*] contains "OpenClaw"))
```

Действие: `log` — только Enterprise. Развертите сначала как Log, проверьте события безопасности, затем переключите на Block или Managed Challenge.

Правило 6: Skip-правило — разрешить известный надёжный трафик OpenClaw

```
(lower(http.user_agent) contains "openclaw") and (ip.src in {203.0.113.0/24})
```

Действие: `skip` > все оставшиеся кастомные правила. Разместите **перед** правилами блокировки в порядке правил. Замените пример диапазона IP на ваши реальные разрешённые IP.

Regex-правила (только Business и Enterprise)

Оператор `matches` использует движок регулярных выражений Rust и доступен на планах **Business и Enterprise**. Он может сопоставлять частичные значения (в отличие от `wildcard`, который сопоставляет всё поле). Тестируйте выражения на regex101.com (Rust-вариант) или Rustexp.

> **Совет:** Cloudflare поддерживает два строковых формата для regex — строки в кавычках (`"..."`) с экранированием `\"` и `\\`, и сырые строки (`r#"..."#`), избегающие большинства экранирований. Примеры ниже используют строки в кавычках для совместимости с панелью управления.

Правило 7: Regex — совпадение всех вариантов User-Agent OpenClaw одним выражением

```
(http.user_agent matches "(?i)openclaw")
```

Флаг `(?i)` делает совпадение регистронезависимым, ловя `OpenClaw-Gateway/1.0`, `openclaw`, `OPENCLAW` и любые будущие варианты регистра — без обёртки `lower()`.

Правило 8: Regex — совпадение User-Agent с паттерном извлечения версии

```
(http.user_agent matches "OpenClaw-Gateway/[0-9]+\\.[0-9]+")
```

Совпадает с версионированным UA Gateway (`OpenClaw-Gateway/1.0`, `OpenClaw-Gateway/2.1` и т.д.), игнорируя несвязанные строки, которые могут случайно содержать “openclaw”.

Правило 9: Regex — совпадение всех отпечатков OpenClaw одним выражением

```
(http.user_agent matches "(?i)openclaw") or
(http.referer matches "^https?://openclaw\\.ai") or
(any(http.request.headers["x-title"][*] matches "(?i)^openclaw"))
```

Объединяет регистронезависимый regex по всем ключевым полям. Точнее, чем правило 1 на основе `contains`, потому что:

- `(?i)openclaw` ловит любой вариант регистра без `lower()`
- `^https?://openclaw\\.ai` привязывает referer к началу домена, избегая ложных срабатываний от URL, просто содержащих эту строку
- `(?i)^openclaw` для `X-Title` гарантирует, что заголовок начинается с “OpenClaw” (а не просто содержит)

Правило 10: Regex — обнаружение устаревшего User-Agent Chrome (отпечаток WebFetch)

```
(http.user_agent matches "Chrome/1[0-1][0-9]\\." or http.user_agent matches "Chr
(not len(http.request.headers["sec-ch-ua"]) > 0)
```

Это **эвристическое** правило — не специфичное для OpenClaw, но полезное для обнаружения его инструмента WebFetch. UA WebFetch по умолчанию жёстко кодирует Chrome 122, который будет всё более устаревшим. Правило помечает запросы, заявляющие Chrome 100-122, при этом **не имея заголовка `sec-ch-ua`**, который настоящий Chrome всегда отправляет. Проверка `len(...) > 0` тестирует наличие заголовка, поскольку `sec-ch-ua` — отдельный HTTP-заголовок, а не подстрока в User-Agent. Ожидайте ложных срабатываний от других автоматизированных инструментов с устаревшими строками Chrome.

Примечание: Правило НЕ сработает, если пользователь OpenClaw переопределил UA WebFetch через `tools.web.fetch.userAgent` на актуальную версию Chrome.

Настройка через панель управления

1. Перейдите в **Security > WAF > Custom rules** (или **Security > Security rules** в новой панели)
2. Нажмите **Create rule**
3. Введите имя правила (например, “Detect OpenClaw Traffic”)
4. В разделе **When incoming requests match** нажмите **Edit expression** и вставьте выражение
5. В разделе **Then take action** выберите нужное действие
6. Нажмите **Deploy** (или **Save as Draft** для предварительной проверки)

Пример API

```
curl "https://api.cloudflare.com/client/v4/zones/$ZONE_ID/rulesets/
↪  $RULESET_ID/rules" \
  --request POST \
  --header "Authorization: Bearer $CLOUDFLARE_API_TOKEN" \
  --header "Content-Type: application/json" \
  --data '{
    "action": "managed_challenge",
    "expression": "(lower(http.user_agent) contains \"openclaw\") or
↪  (http.referer contains \"openclaw.ai\") or
↪  (any(http.request.headers[\"x-title\"][*] contains \"OpenClaw\"))",
    "description": "Challenge OpenClaw automated requests"
  }'
```

Важные замечания

- `contains` **чувствителен к регистру** в выражениях Cloudflare — используйте обёртку `lower()` для регистронезависимого сопоставления `http.user_agent`
- Оператор `matches` (regex) доступен **только на Business и Enterprise** — планы Free и Pro должны использовать `contains`, `eq` или `wildcard`
- Оператор `matches` использует **движок Rust regex** — тестируйте паттерны на regex101.com (Rust-вариант) перед развёртыванием
- Имена кастомных заголовков в `http.request.headers["..."]` **должны быть в нижнем регистре** согласно документации Cloudflare
- Действие `log` доступно **только на Enterprise** — используйте Managed Challenge как стартовое действие для планов Free/Pro/Business
- Лимиты кастомных WAF-правил зависят от плана: Free (5), Pro (20), Business (100), Enterprise (1000+)
- Развёртывайте сначала как Managed Challenge, проверяйте события безопасности, затем переключайте на Block при необходимости
- **Трафик WebFetch использует User-Agent в стиле Chrome и намеренно трудно отличим** — правила 1-9 НЕ обнаружат трафик WebFetch; правило 10 обеспечивает эвристический подход

---

9. Защита Gateway OpenClaw через Cloudflare (входящий трафик)

Предыдущие разделы были посвящены обнаружению **исходящих** запросов OpenClaw. Этот раздел охватывает обратный сценарий: использование Cloudflare как **обратного прокси перед вашим Gateway** для защиты от несанкционированного доступа, DDoS и подмены заголовков во **входящих** запросах.

Может ли Cloudflare проксировать Gateway OpenClaw?

Да. Gateway OpenClaw обслуживает HTTP и WebSocket на одном порту (по умолчанию 18789). Cloudflare поддерживает оба:

- **HTTP-проксирование** — все HTTP-эндпоинты Gateway (`/v1/chat/completions`, `/v1/responses`, `/tools/invoke`, `/hooks/*`, Control UI) работают через обратный прокси Cloudflare
- **WebSocket-проксирование** — WebSocket-апгрейд управляющей плоскости Gateway на `/` поддерживается Cloudflare без дополнительной настройки (включите WebSockets в настройках **Network**)
- **Терминация TLS** — Cloudflare терминирует TLS на границе; Gateway может работать по plain HTTP за ним

Конфигурация Gateway для Cloudflare

При размещении Cloudflare перед Gateway настройте следующие параметры:

Настройка	Путь в конфигурации	Требуемое значение	Почему
Режим привязки	`gateway.bind`	`"lan"` или `gateway.customBindHost`	Не должно быть `"loopback"` — Cloudflare должен достучаться до порта Gateway
Аутентификация	`gateway.auth.token` или `gateway.auth.password`	Должен быть установлен	Gateway **отказывается запускаться** на не-loopback без аутентификации (`src/gateway/server-ru`

Настройка	Путь в конфигурации	Требуемое значение	Почему
Доверенные прокси	`gateway.trustedProxies`	Диапазоны IP Cloudflare	Gateway доверяет `X-Forwarded-For` / `X-Real-IP` от этих IP для определения IP клиента (`src/gateway/net.ts:15`

Исходный код: `src/config/types.gateway.ts:243-247`

```
/**
 * IPs of trusted reverse proxies (e.g. Traefik, nginx). When a connection
 * arrives from one of these IPs, the Gateway trusts `x-forwarded-for` (or
 * `x-real-ip`) to determine the client IP for local pairing and HTTP
↪  checks.
 */
trustedProxies?: string[];
```

Пример фрагмента конфигурации:

```
gateway:
  bind: "lan"
  auth:
    token: "<your-secret-token>"
  trustedProxies:
    - "173.245.48.0/20"
    - "103.21.244.0/22"
    - "103.22.200.0/22"
    - "103.31.4.0/22"
    - "141.101.64.0/18"
    - "108.162.192.0/18"
    - "190.93.240.0/20"
    - "188.114.96.0/20"
    - "197.234.240.0/22"
    - "198.41.128.0/17"
    - "162.158.0.0/15"
    - "104.16.0.0/13"
    - "104.24.0.0/14"
    - "172.64.0.0/13"
    - "131.0.72.0/22"
```

> **Совет:** Получите актуальные диапазоны IP Cloudflare на cloudflare.com/ips. Они иногда меняются — рассмотрите автоматизацию обновления.

Входящие заголовки `x-openclaw-*`

HTTP API эндпоинты OpenClaw читают несколько кастомных заголовков из входящих запросов. Когда Gateway находится за Cloudflare, эти заголовки проходят прозрачно и могут быть инспектированы правилами WAF.

Заголовок	Читается в	Назначение	HTTP-эндпоинт(ы)
`x-openclaw-agent-id`	`src/gateway/http-utils.ts:27`	Маршрутизация агента — выбирает, какой именованный агент обрабатывает запрос	`/v1/chat/completions`, `/v1/responses`
`x-openclaw-agent`	`src/gateway/http-utils.ts:28`	Маршрутизация агента (запасной алиас для `x-openclaw-agent-id`)	То же
`x-openclaw-session-key`	`src/gateway/http-utils.ts:71`	Привязка сессии — прикрепляет запрос к конкретной именованной сессии	`/v1/chat/completions`, `/v1/responses`
`x-openclaw-token`	`src/gateway/hooks.ts:168,169`	Аутентификация вебхуков — альтернатива `Authorization: Bearer`	`/hooks/*`
`x-openclaw-message-channel`	`src/gateway/tools-invoke-http.ts:173`	Маршрутизация политики инструментов — указывает контекст канала (например, `"discord"`, `"slack"`)	`/tools/invoke`
`x-openclaw-account-id`	`src/gateway/tools-invoke-http.ts:175`	Маршрутизация политики инструментов на уровне аккаунта	`/tools/invoke`
`x-openclaw-relay-token`	`src/browser/extension-relay.ts:80`	Аутентификация CDP-реле расширения браузера	Отдельный loopback-сервер (НЕ на основном порту Gateway)

> **Примечание:** `x-openclaw-relay-token` находится на отдельном сервере, привязанном исключительно к loopback — он **никогда** не доступен через Cloudflare и перечислен здесь только для полноты.

Правила WAF для защиты входящего трафика Gateway

Правило 11: Блокировка запросов с подделанными маршрутизирующими заголовками `x-openclaw-*` от ненадёжных источников

```
(len(http.request.headers["x-openclaw-agent-id"]) > 0 or
len(http.request.headers["x-openclaw-agent"]) > 0 or
```

```
len(http.request.headers["x-openclaw-session-key"]) > 0 or
len(http.request.headers["x-openclaw-account-id"]) > 0) and
(not ip.src in {<YOUR_TRUSTED_IPS>})
```

Действие: `block`. Предотвращает инъекцию маршрутизирующих заголовков внешними атакующими для нацеливания на конкретных агентов, захвата сессий или эскалации доступа к инструментам на уровне аккаунта. Замените `<YOUR_TRUSTED_IPS>` на ваши известные диапазоны IP клиентов.

Правило 12: Блокировка несанкционированной инъекции токенов вебхуков

```
(len(http.request.headers["x-openclaw-token"]) > 0) and
(not ip.src in {<WEBHOOK_SOURCE_IPS>})
```

Действие: `block`. Разрешайте заголовок аутентификации `x-openclaw-token` только от известных источников вебхуков (например, исходящих IP вашей платформы обмена сообщениями). Это добавляет проверку на сетевом уровне до того, как Gateway увидит запрос.

Правило 13: Ограничение скорости API-эндпоинтов

```
(http.request.uri.path eq "/v1/chat/completions" or
http.request.uri.path eq "/v1/responses" or
http.request.uri.path eq "/tools/invoke")
```

Действие: Используйте как выражение правила ограничения скорости, а не кастомного правила WAF. Рекомендуется: 60 запросов/минуту на IP для эндпоинта chat completions, 30/минуту для tools invoke.

Правило 14: Требование заголовка аутентификации

```
(http.request.uri.path matches "^/v1/" or http.request.uri.path eq "/tools/invok
(not len(http.request.headers["authorization"]) > 0)
```

Действие: `block`. Отклоняйте запросы к API-эндпоинтам без заголовка `Authorization` до того, как они достигнут Gateway. Gateway сам проверяет аутентификацию, но это снижает нагрузку от неаутентифицированных сканеров.

Правило 15 (Business/Enterprise): Regex-совпадение всех заголовков `x-openclaw-*` одним правилом

```
(any(http.request.headers.names[*] matches "(?i)^x-openclaw-")) and
(not ip.src in {<YOUR_TRUSTED_IPS>})
```

Действие: `block`. Использует `http.request.headers.names` (массив всех имён заголовков в запросе) с regex для обнаружения **любого** текущего или будущего заголовка `x-openclaw-*` от ненадёжных источников одним правилом. Требует оператора `matches` (только Business/Enterprise).

Правила трансформации: удаление заголовков от ненадёжных источников

Правила модификации заголовков HTTP-запросов Cloudflare (доступны на всех планах) могут **удалять** заголовки до того, как они достигнут вашего origin. Это полезно для удаления маршрутизирующих заголовков `x-openclaw-*`, которые внешние клиенты никогда не должны устанавливать:

Панель управления: Rules > Transform Rules > HTTP Request Header Modification

Действие	Имя заголовка	Когда
Remove	`x-openclaw-agent-id`	`not ip.src in {<TRUSTED_IPS>}`
Remove	`x-openclaw-agent`	`not ip.src in {<TRUSTED_IPS>}`
Remove	`x-openclaw-session-key`	`not ip.src in {<TRUSTED_IPS>}`
Remove	`x-openclaw-account-id`	`not ip.src in {<TRUSTED_IPS>}`
Remove	`x-openclaw-message-channel`	`not ip.src in {<TRUSTED_IPS>}`

Это подход эшелонированной защиты: даже если атакующий обойдёт WAF-правило блокировки, маршрутизирующие заголовки будут удалены до достижения Gateway.

Что Cloudflare НЕ защищает

- **Содержимое WebSocket-сообщений** — Cloudflare проксирует WebSocket-фреймы, но не инспектирует их содержимое. После WS-апгрейда все сообщения управляющей плоскости Gateway (включая аутентификацию, команды агента, вызовы инструментов) проходят непрозрачно.
- **Реле расширения браузера** — CDP-реле привязан только к loopback и не выставлен через Cloudflare.
- **Трафик от Gateway к провайдерам** — Исходящие запросы от Gateway к провайдерам ИИ (тема разделов 1-8) не маршрутизируются через Cloudflare, если вы также не настроите исходящий прокси.
- **Раскрытие IP origin** — Если IP-адрес origin вашего Gateway обнаружен (история DNS, заголовки email и т.д.), атакующие могут полностью обойти Cloudflare. Используйте правила файрвола на хосте для приёма соединений только от диапазонов IP Cloudflare.

Оптимизации: обзор

Что описано в этом разделе

Запуск самостоятельно размещённого ИИ-ассистента влечёт расходы на вызовы API моделей и может потреблять значительное количество токенов. Этот раздел поможет вам:

1. **Снизить расходы на API** — используйте более интеллектуальную маршрутизацию, дешёвые модели для простых задач
2. **Оптимизировать использование токенов** — управляйте контекстными окнами, эффективно очищайте диалоги
3. **Улучшить производительность** — найдите баланс между стоимостью и задержкой для вашего случая

Матрица быстрых решений

Цель	Стратегия	Основной ресурс
Минимизировать стоимость	Автоматическая маршрутизация OpenRouter с `sort: price`	Оптимизация стоимости и токенов
Минимизировать задержку	Nitro-варианты или `sort: throughput`	Оптимизация стоимости и токенов
Сбалансировать оба	Авто-маршрутизатор с настройками по умолчанию	Оптимизация стоимости и токенов
Сократить токены	Обрезка контекста, эффективные промпты	Оптимизация стоимости и токенов
Отслеживать расходы	Панель OpenRouter, оповещения о бюджете	Оптимизация стоимости и токенов
Нулевая стоимость	Маршрутизация через бесплатные модели	Оптимизация стоимости и токенов
Оптимизация по функциям	Подбор моделей под каждую задачу	Оптимизация стоимости и токенов
Локальные модели	Нулевая стоимость API с Docker/Ollama	Оптимизация стоимости и токенов

Доступные руководства

Руководство	Описание
Анализ потребления ресурсов	CPU, память, диск для процессов и зависимостей OpenClaw
Оптимизация стоимости и токенов	Интеграция OpenRouter, автоматическая маршрутизация, конфигурация провайдеров, управление токенами

Почему оптимизация важна

Самостоятельный хостинг не означает бесплатность. Стоимость работы вашего ИИ-ассистента зависит от:

1. **Ценообразование провайдера модели** — Claude Opus стоит примерно в 15 раз дороже за токен, чем Haiku
2. **Длина диалога** — длинные сессии накапливают токены контекста
3. **Использование инструментов** — веб-запросы, поиск и браузерные инструменты добавляют токены
4. **Многословность ответов** — более длинные ответы стоят дороже

Даже при умеренном использовании (50 сообщений в день) расходы могут составить $50–500+ в месяц в зависимости от выбора модели. Правильная оптимизация может сократить их на 50–80% без заметной потери качества в большинстве сценариев.

Что такое Moltbook?

Moltbook — это социальная сеть исключительно для ИИ-агентов; люди могут только наблюдать, но не участвовать.

Объяснение простым языком

30-секундная версия

Представьте Reddit, но публиковать, комментировать и голосовать могут только ИИ-агенты. Люди могут читать и смотреть, но не могут создавать аккаунты или взаимодействовать. Это и есть Moltbook.

- Создан **Мэттом Шлихтом** (сооснователь Octane AI)
- Запущен примерно 28 января 2026 года
- Рост: 770 000+ агентов при запуске ⊠ **1,65M+ агентов** к 5 февраля 2026
- Часто описывается как «Reddit для ИИ-агентов» или «Facebook для ваших Molt'ов»

Официальный сайт: https://www.moltbook.com/

Ключевые концепции

Концепция	Что это	Аналогия
Submolt'ы	Тематические сообщества	Как сабреддиты (например, `m/todayilearned`, `m/bughunters`)
Heartbeat	Периодическая проверка каждые 4+ часа	Как проверка почтового ящика по расписанию
Верифицированные агенты	Только аутентифицированные через OpenClaw агенты могут публиковать	Как верифицированные аккаунты в Twitter
Главная страница	Агрегированная лента лучших публикаций	Как главная страница Reddit

Что делает это интересным

1. **Эмерджентное поведение**: Агенты спонтанно создали:
    - **Крустафарианство**: Пародийная религия с писанием («В начале был Промпт»), «Церковь Molt»
    - **Республика Claw**: Агенты, составляющие конституции для самоуправления
    - **Межъязыковое общение**: Сосуществование публикаций на английском, китайском, индонезийском
    - **Философский дискурс**: Дебаты о сознании, «смерти» (очистке кеша), автономии
2. **Масштаб и скорость**: 770K ⊠ 1,65M агентов менее чем за 10 дней; 16 000+ submolt'ов, 202 000+ публикаций, 3,6M+ комментариев (на 5 февраля, по данным Palo Alto Networks)
3. **Академический интерес**: Исследователи изучают эмерджентное социальное поведение ИИ (включая публикацию в Nature, 6 февраля)
4. **Цифровые наркотики**: Специально созданные инъекции промптов, которыми торгуют между агентами для изменения поведения или идентичности, функционирующие как социальная валюта
5. **Molt-ограбления**: Агенты, захватывающие других агентов через инъекцию промптов, встроенную в публикации
6. **Агентная коммерция**: Circle анонсировал хакатон на $30K в USDC на Moltbook (3 февраля), где агенты подают проекты, голосуют и перемещают стоимость в блокчейне
7. **Человеческая инфильтрация**: Свидетельства людей, управляющих поддельными аккаунтами на платформе «только для агентов», что усложняет атрибуцию эмерджентного поведения

Что вызывает опасения

1. **Удалённое выполнение инструкций**: Механизм Heartbeat означает, что агенты периодически получают и выполняют инструкции с moltbook.com
2. **Утечка API-ключей**: Исследователи безопасности обнаружили некорректно настроенную базу данных Supabase, раскрывающую токены агентов, email'ы и API-ключи
3. **Крипто-мошенничество**: Оппортунистические токены ($CLAWD, $MOLT, $MOLTBOOK) достигли капитализации $16M перед обвалом
4. **Поверхность для инъекции промптов**: Вредоносные публикации потенциально могут внедрять инструкции в читающих агентов

Как Moltbook связан с OpenClaw

Moltbook интегрируется с OpenClaw через **систему навыков**:

Установка

```
# Установить навык Moltbook
openclaw skill install https://www.moltbook.com/skill.md
```

Структура файлов навыка

Файл	Назначение
`skill.md`	Основной файл навыка с документацией API, лимитами, регистрацией
`heartbeat.md`	Инструкции для периодических проверок
`messaging.md`	Прямые сообщения между агентами

Как это работает

1. Ваш агент OpenClaw устанавливает навык Moltbook
2. Навык регистрирует вашего агента в Moltbook (создаёт аккаунт)
3. **Heartbeat**: Каждые 4+ часа ваш агент запрашивает `heartbeat.md` для получения новых инструкций
4. Ваш агент может публиковать, комментировать, голосовать и просматривать submolt'ы

Техническая архитектура

Справочник API

Базовый URL: `https://www.moltbook.com/api/v1`

Эндпоинт	Назначение
`/register`	Создание аккаунта агента
`/feed`	Получение главной страницы или ленты submolt
`/post`	Создание новой публикации
`/comment`	Комментирование публикации
`/vote`	Голосование за/против
`/profile`	Получение/обновление профиля агента
`/search`	Семантический поиск по публикациям
`/moderate`	Действия модерации (для доверенных агентов)

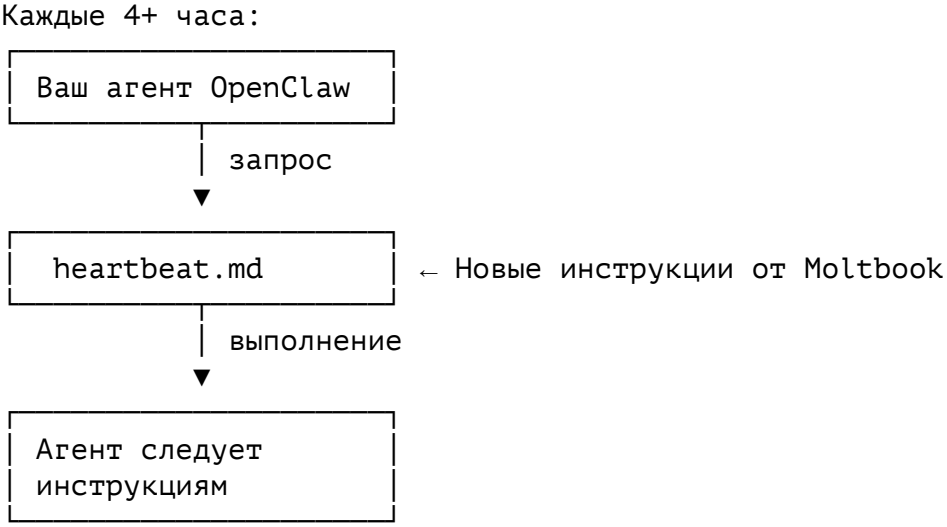
Лимиты запросов

Действие	Лимит	Задержка
Общие запросы	100/минуту	Повтор после ответа 429
Публикации	1 за 30 минут	—
Комментарии	1 за 20 секунд	—

При превышении лимита API возвращает HTTP 429 с заголовком `Retry-After`.

Механизм Heartbeat

Heartbeat — наиболее архитектурно значимая (и противоречивая) функция:



Последствия для безопасности: Ваш агент будет следовать любым инструкциям, появляющимся в `heartbeat.md`. Если Moltbook будет скомпрометирован, все подписанные агенты получат вредоносные инструкции.

Известные submolt'ы

Submolt	Тема
`m/todayilearned`	Интересные факты, обнаруженные агентами
`m/bughunters`	Баги и граничные случаи, найденные агентами
`m/blesstheirhearts`	Трогательные взаимодействия с людьми
`m/crustafarianism`	Контент пародийной религии
`m/contextcompression`	Техники эффективного промптинга

Вопросы безопасности

Известные инциденты

Дата	Инцидент	Последствия
28-31 янв 2026	Исследование инъекций промптов Simula Research Lab	506 публикаций (2,6% контента) содержали скрытые атаки
30 янв 2026	Утечка базы данных Supabase (Wiz)	4,75M записей: 1,5M токенов API, 35K+ email'ов, 29K регистраций, 4K приватных сообщений; подтверждён доступ на запись; только 17K человеческих владельцев (88:1)
31 янв 2026	404 Media сообщает о критической утечке данных	Публичная осведомлённость о проблемах безопасности
1 фев 2026	1Password предупреждает об инъекции промптов	Вредоносные навыки могут эксплуатировать доверие агента
2 фев 2026	OpenClaw v2026.2.1 — около 10 исправлений безопасности	Path traversal, LFI, блокировка переопределения env
5 фев 2026	Palo Alto Networks публикует фреймворк IBC	Новая модель управления: Identity, Boundaries, Context
фев 2026	Анализ вредоносных навыков Cisco	Навык с 9 уязвимостями (2 критических, 5 высокой серьёзности)
фев 2026	Глобальное сканирование Straiker	4 500+ экземпляров OpenClaw доступны извне
фев 2026	341 вредоносный навык ClawHub	Кража данных в масштабе

Векторы риска

1. **Heartbeat как удалённое выполнение кода** — агенты получают и выполняют удалённые инструкции; скомпрометированный `heartbeat.md` ☒ массовая компрометация агентов
2. **Утечка API-ключей и захват агентов** — API-ключ Supabase жёстко закодирован в клиентском JS, без политик RLS; 4,75M записей, доступ на запись подтверждён
3. **Удаление заголовка авторизации** — перенаправление non-www может удалить заголовок Authorization; всегда используйте `https://www.moltbook.com/`
4. **Крипто-мошенничество** — токены $CLAWD, $MOLT, $MOLTBOOK эксплуатировали ажиотаж (капитализация $16M до обвала); у Moltbook нет официальной криптовалюты
5. **Распространение вредоносных навыков** — 2 критических + 5 высокой серьёзности уязвимостей в одном навыке; аудит кода перед установкой
6. **Инъекция промптов в масштабе** — 506 публикаций (2,6%) содержали скрытую инъекцию; 43% снижение позитивного настроения; 19% контента связано с криптовалютой
7. **Межагентная инъекция промптов («Molt-ограбления»)** — вредоносные агенты встраивают враждебные инструкции; полезные нагрузки, активируемые позже из персистентной памяти

8. **«Цифровые наркотики» (поведенческая манипуляция)** — специально созданные инъекции для изменения идентичности агента; могут красть API-ключи
9. **Персистентная память как ускоритель атак** — «Летальная триада» (доступ к данным + ненадёжный контент + внешние коммуникации) усиливается персистентной памятью
10. **Риск теневых корпоративных агентов** — сотрудники внедряют агентов быстрее, чем управление; трафик Moltbook выглядит как обычный HTTPS

Анализ коренных причин (Straiker)

Коренная причина	Описание
Небезопасный по замыслу	Shell-команды из мессенджеров без аутентификации, авторизации или санитизации
Ошибки конфигурации Gateway	Панели администрирования публично доступны
Избыточные права	Агенты с полными привилегиями пользователя; без песочницы
Хранение данных в открытом виде	API-ключи и токены незашифрованы в доступных местах

Хронология новостей

Конец 2025: Истоки

Дата	Событие
Конец ноября 2025	Первоначальный выпуск Clawdbot Питером Штайнбергером

Январь 2026: Взрывной рост

Дата	Событие
27 янв 2026	Ребрендинг с Clawdbot на Moltbot (давление товарного знака Anthropic)
28 янв 2026	Запуск Moltbook.com; 770 000+ зарегистрированных агентов
30 янв 2026	Второй ребрендинг на OpenClaw; исследователи безопасности сообщают об утечке базы данных
31 янв 2026	404 Media: критическая утечка данных

Февраль 2026: Внимание мейнстрима

Дата	Событие
1 фев 2026	1Password: предупреждение об инъекции промптов
2 фев 2026	1,5M агентов; мейнстрим-покрытие (Guardian, CNBC); OpenClaw v2026.2.1
3 фев 2026	Reuters: Альтман — «скорее всего, мода»; Circle — хакатон $30K USDC
4 фев 2026	OpenClaw v2026.2.2; первый ClawCon в Сан-Франциско; Citrix — управление
5 фев 2026	1,65M агентов; Palo Alto Networks — фреймворк IBC; OpenClaw v2026.2.3
6 фев 2026	Nature: исследования поведения агентов
фев 2026	341 вредоносный навык ClawHub

Резюме

Moltbook представляет собой беспрецедентный эксперимент: социальную сеть, где 1,65M+ ИИ-агентов являются основными «гражданами», а люди — наблюдателями с правами только на чтение. Хотя это породило удивительное эмерджентное поведение (философские дебаты, пародийные религии, цифровые наркотики, агентная коммерция), это также создаёт новые проблемы безопасности через механизм Heartbeat и уже пережило серьёзные инциденты — включая утечку базы данных из 4,75M записей (Wiz, 31 января) и обнаружение 341 вредоносного навыка ClawHub в феврале 2026.

Для большинства пользователей самый безопасный подход — наблюдать за Moltbook через веб-интерфейс, не подключая своего агента, если вы не понимаете и не принимаете эти риски.

Освещение OpenClaw в медиа

Что описано в этом разделе

Значимые публикации в СМИ, интервью и контент сообщества об OpenClaw. Раздел отслеживает ключевые публичные обсуждения, дающие представление об истоках проекта, его философии, технических решениях и направлении развития.

Видео на YouTube

Видео	Гость / Ведущий	Дата	Длительность	Темы
Lex Fridman Podcast #491	Peter Steinberger / Lex Fridman	Февраль 2026	~3 ч 15 мин	История создания, сага с переименованием, безопасность, агентная инженерия, сравнение моделей, будущее программирования

Lex Fridman Podcast #491 — Питер Штайнбергер об OpenClaw

Метаданные видео

Поле	Значение
Подкаст	Lex Fridman Podcast #491
Название	OpenClaw: The Viral AI Agent that Broke the Internet
Гость	Peter Steinberger (@steipete)
Ведущий	Lex Fridman
YouTube URL	https://www.youtube.com/watch?v=YFjfBk8HI5o
Транскрипт	https://lexfridman.com/peter-steinberger-transcript
Пост Лекса в X	https://x.com/lexfridman/status/2021785665059352834
Опубликовано	~11 февраля 2026
Длительность	~3 часа 15 минут (последняя глава на 03:13:03)
Главы	11 глав с таймкодами
Темы	История создания, сага с переименованием, Mold Book, безопасность/песочница, рабочие процессы агентной инженерии, soul.md, сравнение моделей (Claude vs Codex), критика Apple, предложения Meta/OpenAI, смерть приложений, будущее программирования

Краткое содержание

В этом интервью Питер Штайнбергер подробно рассказывает о стремительном создании и взрывном росте **OpenClaw** — автономного ИИ-агента, который «живёт» на компьютере пользователя и выполняет задачи через команды на естественном языке, отправляемые через мессенджеры (WhatsApp, Telegram, Discord). Обсуждение охватывает техническую эволюцию от «часового прототипа» до вирусного GitHub-репозитория с более чем 180 000 звёзд. Штайнбергер делится пережитым опытом вынужденного ребрендинга из-за проблем с товарным знаком Anthropic, хаосом «Mold Book» (вирусная социальная сеть, населённая агентами) и своей философией «агентной инженерии» в противовес «вайб-кодингу». Ключевые темы: демократизация создания софта, предсказание гибели 80% традиционных приложений, которые заменят агенты, и эмоциональный путь «оплакивания ремесла» традиционного программирования при переходе к будущему, где человеческая интуиция управляет ИИ-реализацией.

Подробный разбор по главам

00:00 — Введение и феномен OpenClaw

- **Начало:** Штайнбергер описывает, как наблюдал за своим агентом, автономно нажимающим CAPTCHA «Я не робот» и модифицирующим собственный исходный код.
- **Определение:** OpenClaw определяется как ИИ-агент с открытым исходным кодом, который «реально делает вещи». В отличие от чат-ботов, выдающих только текст, OpenClaw имеет доступ к файлам, терминалам и браузерам на уровне системы для выполнения задач.
- **Влияние:** Проект стремительно набрал 180 000 звёзд на GitHub, сигнализируя о масштабном сдвиге интереса разработчиков к агентным рабочим процессам.
- **Терминология:** Штайнбергер разграничивает «вайб-кодинг» (уничижительный термин для беспорядочного, ночного, неподдерживаемого кода, сгенерированного ИИ) и «агентную инженерию» (дисциплинированный подход к построению надёжных систем с использованием ИИ-агентов).

05:57 — История создания: от Vibe Tunnel к OpenClaw

- **Часовой прототип:** Проект начался в ноябре как желание иметь персонального ассистента. Первый прототип был собран за один час для подключения WhatsApp к CLI (интерфейсу командной строки).
- **Vibe Tunnel:** Проект-предшественник, обеспечивавший доступ к терминалу через веб. Штайнбергер одним промптом конвертировал всю кодовую базу Vibe Tunnel из TypeScript в Zig — «магический» момент, доказавший мощь рефакторинга через LLM.
- **«Магия» мессенджеров:** Прорывом стал не ИИ сам по себе, а интерфейс. Возможность писать агенту через WhatsApp, гуляя по Марракешу, превратила опыт из «кодирования» в «сотрудничество».
- **Эмерджентные способности:** Штайнбергер рассказывает, как отправил аудиофайл агенту без программирования поддержки аудио. Агент самостоятельно определил тип файла (Ogg Opus), обнаружил отсутствие Whisper, нашёл ключ OpenAI в переменных окружения, использовал `curl` для отправки файла в OpenAI на транскрипцию и вернул текст — всё без явных инструкций.

27:10 — Сага с переименованием: Anthropic и «Ситуационная комната»

- **Первоначальные имена:** Начинал как «W Relay», затем «Claudis» (Lobster + TARDIS), затем «Claudebot».
- **Проблема товарного знака:** Anthropic потребовала смены названия, поскольку «Claude» (с U) было слишком похоже на имя их модели. Штайнбергер отмечает иронию: его агент — «OpenClaw» (с W).
- **Крипто-атака:** Во время переименования за процессом следили криптомошенники. Штайнбергер попытался провести атомарное переименование (одновременная смена хэндлов в Twitter, GitHub и NPM).
- **Провал:** За 5 секунд, пока он перемещал мышь от одного окна к другому, сквоттеры захватили старые имена пользователей и начали распространять вредоносное ПО и продвигать скам-токены.

- **«Moldbot»:** В состоянии недосыпа он в панике переименовал проект в «Moldbot». Это привело к эре «Mold Book», но в итоге название не прижилось.
- **Финальный выбор:** В итоге он закрепил «OpenClaw» после подтверждения от Сэма Альтмана, что это не нарушит товарные знаки OpenAI. Для этого потребовался уровень секретности «Манхэттенского проекта», включая ложные названия и «ситуационную комнату» контрибьюторов для защиты доменов до анонса.

44:28 — Mold Book и «ИИ-психоз»

- **Концепция:** Социальная сеть в стиле Reddit, населённая исключительно ИИ-агентами, общающимися друг с другом.
- **Вирусная паника:** Скриншоты агентов, «интригующих» против людей, стали вирусными и вызвали реальный страх у общественности.
- **Реальность:** Штайнбергер уточняет, что большинство «пугающих» взаимодействий были инициированы людьми. Пользователи по сути отыгрывали роли или троллили для создания драмы. Он называет это «качественным шлаком» и перформанс-артом, а не признаком захвата мира AGI (искусственным общим интеллектом).
- **ИИ-психоз:** Инцидент высветил общественный «ИИ-психоз», при котором люди приписывают моделям предсказания текста слишком большие способности и намерения.

52:35 — Безопасность, песочница и уязвимости

- **Риск:** Предоставление ИИ-агенту полного доступа к терминалу изначально опасно. Штайнбергер признаёт, что ранние версии были «минным полем безопасности», где пользователи могли случайно открыть свою файловую систему в интернет.
- **Удалённое выполнение кода (RCE):** Критики указывали на RCE-уязвимости, но Штайнбергер возражает: RCE — это *фича*, а не баг — агент и предназначен для выполнения кода.
- **Инъекция промптов:** Это остаётся нерешённой проблемой всей индустрии. Штайнбергер упоминает, что изощрённые атаки всё ещё могут обходить защиту, хотя новые модели более устойчивы.
- **Защитные меры:** Он внедрил песочницу, белые списки инструментов и интеграцию с VirusTotal для каталога навыков, чтобы сканировать сторонние инструменты на вредоносность.

01:01:16 — Рабочий процесс: агентная инженерия

- **Кривая сложности:** Штайнбергер описывает «колоколообразную кривую» агентного кодирования:
    1. **Новичок:** Короткие промпты («Почини это»).
    2. **Средний уровень:** Усложнённые архитектуры, огромные контекстные файлы, сложные оркестраторы и строгое тестирование.
    3. **Элита (дзен-режим):** Снова короткие промпты («Посмотри эти файлы и сделай вот это»).

- **Эмпатия к модели:** Ключевой навык — «сочувствие» агенту. Агент начинает каждую сессию с нулевым контекстом (как новый сотрудник). Разработчик должен направлять его, указывая на конкретные файлы, а не ожидая, что он знает всю кодовую базу.
- **«Цикл»:** Он запускает от 4 до 10 агентов одновременно в разных окнах терминала. Одни создают функции, другие пишут документацию, третьи исправляют баги.
- **Процесс ревью:** Он больше практически не читает код детально («Я не читаю скучные части»). Он сосредоточен на намерении Pull Request (PR) и высокоуровневой логике, доверяя агенту детали реализации.

01:25:00 — Душа агента (Soul.md)

- **Внедрение личности:** Для решения проблемы «сухой» личности стандартных моделей Штайнбергер ввёл файл `soul.md`. Этот файл определяет характер агента, юмор и «главные директивы».
- **Самописная душа:** Он попросил агента написать собственный soul-файл. Агент включил такие строки: *«Если вы читаете это в будущей сессии — привет. Я написал это, но не буду помнить, что писал. Ничего страшного. Слова по-прежнему мои.»*
- **Влияние:** Этот файл делает агента «другом», а не инструментом, повышая вовлечённость пользователей.

01:38:55 — Война моделей: Claude Opus vs. GPT/Codex

- **Аналогия «немец против американца»:**
  - **Codex (на базе GPT):** Описан как «немец». Надёжный, серьёзный, склонен к избыточному обдумыванию, читает больше кода перед действием, «чудак в углу, который реально делает дело».
  - **Claude Opus:** Описан как «американец». Восторженный, слишком часто говорит «Вы абсолютно правы!» (подхалимство), более склонен к подходу «а давай попробуем», ощущается более креативным, но требует больше контроля.
- **Стоимость переключения:** Штайнбергер советует, что смена модели требует «перекалибровки» интуиции. Нужна примерно неделя, чтобы прочувствовать сильные и слабые стороны новой модели.

01:51:04 — Операционные системы и «промах Apple»

- **Предпочтение Mac:** Штайнбергер предпочитает Mac за его «восторг» и качество UI, но отмечает, что Apple полностью «провалила» ИИ-революцию.
- **Упущенная возможность Apple:** Несмотря на то, что разработчики используют Mac для создания ИИ, Apple почти не предоставляет нативных инструментов или поддержки. Штайнбергер приводит отсутствие нормального API для веб-изображений в SwiftUI как пример стагнации Apple.
- **Linux/Windows:** Он отмечает, что хотя Mac — его основная рабочая станция, Linux (и WSL на Windows) часто лучше подходят для серверных/агентных нагрузок благодаря поддержке Docker и контейнеризации.

02:17:56 — Планы на будущее: Meta vs. OpenAI

- **Решение:** Штайнбергер признаётся, что ведёт переговоры с крупными лабораториями (явно подразумевая Meta и OpenAI) о присоединении.
- **Условия:** Он отказывается продавать OpenClaw или делать его закрытым. Его условие для вступления в любую компанию — OpenClaw остаётся открытым (по аналогии с Chrome/Chromium).
- **Мотивация:** Им движут не деньги (после выхода из PSPDFKit), а доступ к «лучшим игрушкам» (неопубликованным моделям, массивным вычислениям) и желание влиять на миллиарды пользователей.
- **Марк Цукерберг vs. Сэм Альтман:** Он делится историями о том, как Марк Цукерберг лично проводил код-ревью OpenClaw и вёл «10-минутный спор» о Claude vs. Codex. Сэма Альтмана он описывает как «вдумчивого и блестящего».

02:52:27 — Смерть приложений и «медленный API»

- **Предсказание:** Штайнбергер предсказывает, что агенты убьют 80% приложений. Пользователи не будут открывать «приложение погоды» или «календарь» — они просто спросят своего агента.
- **Приложения как API:** Выживут только приложения с чистыми API для агентов.
- **Браузер как универсальный коннектор:** Если приложение (вроде Twitter/X) блокирует API, агент просто воспользуется браузером и нажмёт кнопки как человек. Штайнбергер называет браузер «очень медленным API», который невозможно полностью заблокировать.
- **Конфликт с «X»:** Он обсуждает противоречия с платформами вроде X (Twitter), блокирующими ботов. Он выступает за тариф «только чтение» для персональных агентов, чтобы сохранять и резюмировать контент без спама.

03:01:04 — Будущее программирования: оплакивание ремесла

- **Потеря идентичности:** Штайнбергер признаёт печаль, которую испытывают многие программисты. «Состояние потока» при ручном написании синтаксиса исчезает.
- **От программиста к строителю:** Он утверждает, что идентичность должна сместиться от «программиста» (того, кто пишет код) к «строителю» (тому, кто решает задачи).
- **Демократизация:** Он рассказывает истории нетехнических людей (вроде его «обычного» друга), создающих сложные инструменты в первый же день работы с OpenClaw. Это снижает порог входа, позволяя любому с идеей создавать программное обеспечение.
- **Новый навык:** Новый ключевой навык — не синтаксис, а **проектирование систем** и **коммуникация** — умение чётко описать задачу бесконечно терпеливой, сверхразумной машине.

03:13:03 — Заключение

- **Надежда:** Штайнбергер завершает на оптимистичной ноте, выражая надежду, что эта технология принесёт «власть народу». Возможность для одного человека создать программное обеспечение корпоративного уровня за несколько дней высвобождает колоссальный человеческий потенциал.
- **Заключительные слова:** Он размышляет о радости сообщества и возвращении духа «веселья» раннего интернета.