🚀 9 инструментов для настройки Claude Code

🛠️ 9 инструментов для превращения Claude Code в профессиональную среду разработки

Ключевые тезисы:

- Без настройки Claude Code работает на 20-30% своих возможностей.
- Представленные инструменты бесплатны, их установка занимает 20-30 минут.
- Каждый инструмент закрывает конкретную проблему: анализ проекта, правила, память, тесты, мониторинг расходов, аудит кода.
- Не нужно ставить всё сразу. Начинать следует с `claude.md` и добавлять инструменты постепенно.

🎯 Об авторе и подходе

Эдвард Гришин, основатель Futura AI, с 12-летним опытом в IT/e-commerce (Яндекс Маркет, Ozon, Технопарк). Claude Code — его основной рабочий инструмент. Все рекомендации основаны на реальном опыте и бизнес-задачах.

🔍 Анализ проекта: Claude Code Setup

Официальный плагин от Anthropic для анализа структуры проекта.

- **Что делает:** Сканирует файлы, зависимости, конфиги и даёт рекомендации (какие хуки поставить, какие MCP-серверы нужны).
- **Формат:** Read-only, ничего не меняет.
- **Плюсы:** Официальный, безопасный, быстрый (10-30 секунд).
- **Минусы:** Только рекомендации, не настраивает автоматически.

- **Вывод:** Первый шаг в любом проекте. Запустите `claude setup`, чтобы увидеть, что можно улучшить.

📜 Правила проекта: Claude.md + Carpati Skills

Файл `claude.md` в корне проекта — это «конституция проекта», которую Claude читает при каждой сессии.

Принципы Андрея Карпати (экс-директор AI в Tesla, сооснователь OpenAI):

1. **Think before coding:** Озвучить допущения и план перед написанием кода.
2. **Simplicity first:** Не добавлять лишнего, не создавать избыточных абстракций.
3. **Surgical changes:** Вносить минимальные правки, не переписывать работающий код.
4. **Goal-driven:** Определять конкретные критерии успеха (например, «endpoint возвращает 200»).

Как создать `claude.md`:

- Используйте встроенную команду `/init` в Claude Code.
- Claude задаст вопросы о стеке, правилах, тестировании и развернёт готовый файл.
- **Важно:** Если файл уже есть, не перезаписывайте его. Попросите Claude скачать файл Карпати с GitHub и интегрировать полезные правила в существующий `claude.md`.

Плюсы: Ноль зависимостей, работает с любой версией Claude Code.
Минусы: Это просто промпт, а не инструмент. В длинных сессиях (100+ сообщений) Claude может забывать правила.
Совет: Оптимальный размер `claude.md` — 100-200 строк. Детали выносите в ссылки на другие файлы.

🧠 Постоянная память между сессиями: Agent Memory

Решает проблему потери контекста при закрытии и открытии новой сессии.

Четыре уровня памяти:

1. **Рабочая:** Текущие задачи и контекст.
2. **Эпизодическая:** Что было в прошлых сессиях (вчера нашёл баг в интеграции).
3. **Семантическая:** Знания о проекте (используется FastAPI + PostgreSQL 16).
4. **Процедурная:** Как делать вещи (деплой через Docker Compose).

Технические особенности:

- Тройной поиск: по точным словам (BM25), по смыслу (векторный), по связям (граф знаний).
- Точность 95,2% (из 5 результатов 4,7 попадают в точку).
- Всё локально (SQLite), данные никуда не отправляются.
- Кросс-агентность (работает через MCP).

Альтернатива: Claude Mem (более популярен, 68.9k звёзд на GitHub).
Выбор автора: Agent Memory выигрывает по точности, наличию графа знаний и кросс-агентности.

🔗 Работа с GitHub и актуальная документация

Это связка двух инструментов.

1. GitHub CLA (Code Link Access)

- **Вопрос:** В чём разница с GitHub MCP?
- **GitHub CLA:** $3.20 за 10k операций, 100% надёжность, нативный, не требует MCP.
- **GitHub MCP:** $55.2 за 10k операций (в 17 раз дороже), 72% надёжности, 43 инструмента занимают 28k токенов контекста.
- **Вывод:** GitHub MCP нужен только enterprise-командам с кастомными workflow. Для всех остальных — используйте GitHub CLA (он уже есть по умолчанию, нужно только подключить аккаунт).

2. Актуальная документация библиотек
Проблема: Claude обучен на данных до определённой даты и может выдумывать или использовать устаревшие API.

- **Context7:** Бесплатный, 1000 запросов в месяц, точность ~65%.
- **Deepseek:** Платный, точность ~90%, экономит токены контекста.
- **Вывод:** Для старта обязателен Context7. Deepseek — если готовы плавать для экономии токенов.

🤔 Визуализация мышления: Sequential Thinking MCP

Официальный инструмент Anthropic для визуализации процесса рассуждений Claude.

- **Проблема:** Extended Thinking (внутренний процесс Claude) невидим. Вы получаете только финальный ответ.
- **Sequential Thinking:** Делает мышление видимым. Claude разбивает задачу на шаги, вы видите каждый шаг, ветвления, возможность вернуться назад.
- **Использование:** Не для простых задач (бесполезно). Для архитектурных решений, сравнения вариантов (например, 3 способа интеграции CRM), сложных дебагов.
- **Плюсы:** Видимость, ветвление, возможность вмешаться на любом этапе.
- **Минусы:** Тратит больше токенов.
- **Вывод:** Для сложных архитектурных решений — must-have. Используется несколько раз в неделю.

🛡️ Защита от ошибок и экономия денег

Инструмент 6: TDD Guard
Хук, который **физически блокирует** запись кода, если нет падающего теста.

- **Три правила:** 1) Сначала падающий тест, потом код. 2) Реализация не должна превышать требования теста. 3) Один тест — один цикл.
- **Поддержка:** JavaScript, TypeScript, Python, PHP, Go.
- **Как работает:** Claude начинает писать код → TDD Guard говорит «Стоп, сначала тест» → Пишется тест (падает) → Пишется минимальный код → Тест проходит → Рефакторинг.
- **Плюсы:** Жёсткий enforcement TDD, обойти нельзя (только отключить), лёгкий.
- **Минусы:** Замедляет разработку, не для прототипов.

- **Альтернатива:** Superpowers TDD Skill (только рекомендует, можно проигнорировать).
- **Вывод:** Для серьёзных проектов с высокими требованиями к качеству. Для прототипов можно отключать.

Инструмент 7: Контроль расходов (CC Usage)
Показывает, куда и сколько уходят токены.

- **Проблема:** Claude Code, особенно Opus 4.7, потребляет много токенов. Длинная сессия может исчерпать лимит к обеду.
- **Что делает:** Дневные/месячные отчёты, Jupyter-экспорт, трекинг кэша, MCP-интеграция (можно спросить Claude о расходах прямо в сессии).
- **Плюсы:** Зрелый проект, работает оффлайн, MCP-интеграция.
- **Минусы:** Только CLI, нет GUI и предсказаний.
- **Альтернативы:** Claude Code Usage Monitor (реал-тайм мониторинг + ML-предсказания), Claude Usage Dashboard (визуальный дашборд в браузере).
- **Главный совет:** Используйте команду `/clear` после каждого логического блока работы для сброса контекста и экономии токенов.

Инструмент 8: Аудит технического долга (Tech Debt Skill)
Скилл для полного аудита кодовой базы по 9 измерениям. Каждая проблема привязана к конкретному файлу и строке.

9 измерений аудита:

1. Архитектурный распад (нарушение паттернов, циклические зависимости).
2. Типобезопасность (any в TypeScript, отсутствие type hints в Python).
3. Покрытие тестами.
4. Зависимости (устаревшие версии, известные уязвимости).
5. Безопасность (SQL-инъекции, отсутствие rate-limiting).
6. Дублирование кода.
7. Мёртвый код.
8. Конфигурация и секреты (хардкоженные ключи, неправильный .env).
9. Документация (отсутствующая или устаревшая).

- **Плюсы:** Конкретные рекомендации с указанием файла и строки.
- **Минусы:** Первый запуск 5-20 минут, потребляет токены на больших базах, нет CI/CD интеграции (только ручной запуск).
- **Вывод:** Запускать раз в месяц или перед крупным релизом. Лучше найти проблемы через аудит, чем от реальных пользователей.

🎯 Итоговый список инструментов

1. **Claude Code Setup:** Анализ проекта при первом подключении.
2. **Claude.md + Carpati Skills:** Правила для каждой сессии.
3. **Agent Memory (или аналог):** Постоянная память между сессиями.
4. **GitHub CLA:** Нативная работа с GitHub (без MCP).
5. **Context7 / Deepseek:** Актуальная документация библиотек.
6. **Sequential Thinking:** В
