# Разработка с ИИ

## ОТ ИДЕИ ДО ПРОДАКШЕНА

Полное универсальное руководство по созданию собственного сайта, приложения или стартапа с нуля. Освойте ИИ-агентов и инструменты, подключите платежи, разверните свой сервер и изучите маркетинговые приемы для органического роста.

apifreellm.com

Версия 1.0 — Февраль 2026

# Содержание


## 1. Введение

Мир изменился

Зачем создан этот курс

## 2. Стек разработки с приоритетом ИИ

Обзор ИИ-агентов

Выбор и настройка агента

Необходимые инструменты: Git и GitHub CLI

## 3. Ваша первая сборка: от нуля до запуска

Никогда не начинайте с нуля

Выбор правильного стека

## 4. От кода до продакшена

Платежи через Stripe

Хостинг: AWS, Hetzner и другие

Cloudflare: производительность и защита

CI/CD с GitHub Actions

## 5. Маркетинг и SEO-тактики

Стратегии органического роста

SEO, контент и дистрибуция

## Бонус: готовые исходные коды

# 1. Введение

> *Вы читаете это прямо сейчас, потому что где-то, каким-то образом, комбинация маркетинговых стратегий, SEO-техник и грамотного позиционирования привела этот курс на ваш экран. Уже одно это должно вам кое-что сказать: тактики из этого руководства действительно работают. И да, вы изучите каждую из них.*

## Мир изменился

Разработка программного обеспечения уже не та, что раньше. Еще несколько лет назад создание веб-приложения требовало месяцев работы, команды разработчиков и значительного бюджета. Сегодня один человек с правильными инструментами и правильными знаниями может создать и запустить готовое к продакшену приложение за дни, а не за месяцы.

Этот курс написан профессионалами, которые работают в индустрии и ежедневно используют агентные ИИ-инструменты для создания реальных продуктов. Мы не преподаем теорию из учебника. Мы учим тому, что реально работает в реальном мире, прямо сейчас.

## Зачем создан этот курс

ИИ и LLM сейчас являются более качественными и быстрыми исполнителями, чем люди. Они могут писать код, отлаживать его, рефакторить и деплоить со скоростью, с которой ни один человек не может сравниться. Но есть то, чего им фундаментально не хватает: **идей**.

> ИИ-модели — выдающиеся исполнители, но не новаторы. Они не видят пробел на рынке и не думают: «Я мог бы создать что-то, чтобы это исправить». Это ваша работа. Ваша роль — быть архитектором идей. ИИ — ваш строитель.

Когда вы даете LLM плохие инструкции, она все равно что-то выдаст. Она не остановится, чтобы объяснить, что на самом деле было бы лучше. Качество того, что вы создаете, прямо пропорционально качеству ваших знаний. Этот курс устраняет этот пробел.

Это не просто курс по разработке. Это полный план действий для перехода от идеи к работающему продукту, приносящему доход. Включая крайне эффективные маркетинговые и SEO-тактики — те самые техники, которые привели вас на эту самую страницу.

# 2. Стек разработки с приоритетом ИИ

> *Инструменты, которые вы выберете, определят, насколько быстро вы будете двигаться. В этой главе мы разберем доступные сегодня ИИ-агенты для кодинга, поможем вам выбрать подходящий и научим стратегиям промптинга, которые отличают любителей от профессионалов.*

> **Примечание:** Это руководство написано на примере Windows, но все инструменты и шаги одинаково работают на macOS и Linux. Google Antigravity, Claude Code и все остальные приложения, рассмотренные в этом курсе, полностью кроссплатформенны. Если вы на Mac или Linux, просто следуйте тем же шагам — интерфейсы и процессы одинаковы.

## Обзор ИИ-агентов

Рынок ИИ-инструментов для кодинга буквально взорвался. Но не все инструменты одинаковы. Некоторые — по сути, продвинутое автодополнение. Другие — полноценные автономные агенты, которые могут читать всю вашу кодовую базу, планировать стратегию, вносить изменения в несколько файлов, запускать тесты и исправлять ошибки самостоятельно. Понимать разницу — критически важно.

Есть две фундаментальные категории ИИ-инструментов для кодинга:

- **Встроенные помощники** — Они находятся внутри вашего редактора и предлагают код по мере ввода. Вспомните оригинальный GitHub Copilot. Они реактивны: ждут, пока вы напишете, а затем пытаются угадать, что идет дальше. Полезны, но ограничены.
- **Агентные инструменты** — Это совершенно другой уровень. Вы даете им задачу на естественном языке, и они автономно планируют, пишут, редактируют, отлаживают и итерируют. Они не просто предлагают строку кода. Они создают фичи. Именно здесь настоящая сила.

Вот инструменты, которые сейчас действительно важны:

### Claude Code (от Anthropic)

Claude Code — это, по нашему опыту, самый мощный ИИ-агент для кодинга, доступный сегодня. Это терминальный агент, который работает непосредственно в директории вашего проекта. Вы даете ему инструкции на естественном языке, и он читает ваши файлы, пишет код, выполняет команды, создает коммиты и исправляет ошибки автономно. Он имеет полный доступ к вашей файловой системе и оболочке, что делает его невероятно эффективным для реальной разработки. Вы можете установить его через npm (`npm install -g @anthropic-ai/claude-code`) или использовать как расширение VS Code, о чем мы поговорим чуть позже.

### Google Antigravity

Antigravity — это среда разработки от Google, которая привносит ИИ-чат и агентные возможности прямо в ваш рабочий процесс. Представьте его как интеллектуальное рабочее пространство, где вы можете взаимодействовать с ИИ-агентами через чат-интерфейсы, а из каждого разговора с агентом можно открыть интегрированный редактор VS Code. Это ключевой момент: Antigravity дает вам разговорный ИИ-слой, а VS Code обеспечивает мощность редактирования кода. Сочетание безупречно — вы обсуждаете, что создавать, с агентом, а затем сразу переходите к коду без переключения окон.

### Cursor

Cursor — это форк VS Code с глубоко интегрированным ИИ. У него есть как встроенные подсказки, так и агентный режим «Composer», где вы можете описывать изменения в нескольких файлах. Интерфейс будет знакомым, если вы уже используете VS Code, и он поддерживает несколько моделей (Claude, GPT и др.). Это солидный инструмент, особенно если вы предпочитаете полностью визуальный рабочий процесс. Однако его агентные возможности, хотя и хороши, не так автономны, как у Claude Code. Вам часто придется просматривать и одобрять отдельные изменения шаг за шагом.

> Наша рекомендуемая конфигурация: **Google Antigravity + Claude Code как расширение VS Code.** Используйте агентные чаты Antigravity для планирования и обсуждения работы, затем откройте интегрированный VS Code и позвольте Claude Code выполнять тяжелую работу. Это дает вам лучшее из обоих миров: интеллектуальный диалог для планирования и автономное выполнение кода для строительства.

## Выбор и настройка агента

Установить и запустить агентный инструмент — это простая часть. Получить от него максимум — это уже требует понимания того, как он работает, выбора правильной подписки и правильной настройки.

### Подписка Claude: начните с Pro

Claude Code требует аккаунт Anthropic. Мы рекомендуем начать с **подписки Claude Pro**, которая является начальным платным тарифом. Она доступна по цене и дает вам доступ к Claude Code со всеми его функциями. Это идеальная отправная точка, чтобы поэкспериментировать, освоить рабочий процесс и создать свой первый простой проект.

Однако учтите, что тариф Pro имеет **ограниченное использование**. Если вы интенсивно используете Claude Code, вы относительно быстро достигнете лимита. Для обучения и небольших проектов его более чем достаточно. Но если вы уже знаете, что хотите использовать Claude Code как основной инструмент разработки и планируете работать с ним по несколько часов каждый день, рассмотрите переход на один из более высоких тарифов (например, Max) с самого начала. Более высокие тарифы предлагают значительно больше использования, что означает меньше прерываний и более плавный рабочий процесс при создании крупных проектов.

### **Модель**: Claude Opus 4.6

Мы в настоящее время рекомендуем использовать **Claude Opus 4.6** в качестве основной модели разработки. Это, на данный момент, лучшая модель для создания сайтов и приложений. Она понимает сложные архитектуры, пишет чистый и готовый к продакшену код, обрабатывает изменения в нескольких файлах с замечательной точностью и редко нуждается в исправлениях с первой попытки. При работе с Claude Code установите Opus 4.6 как модель по умолчанию. Разница в качестве вывода по сравнению с меньшими моделями заметна сразу.

### **Настройка** Antigravity

С этого момента руководство продолжается с использованием **Google Antigravity** в качестве нашей основной среды разработки. Вот как все подготовить.

**Шаг 1: Создайте папку проекта.** Перед открытием Antigravity создайте папку на вашем компьютере, где будет жить ваш первый проект. Например, на Windows вы можете создать `C:\Projects\my-first-app`, или на Mac/Linux `~/Projects/my-first-app`. Это папка, которую Antigravity откроет как рабочее пространство. Пока она может быть пустой — мы наполним ее кодом позже в курсе.

**Шаг 2: Установите и запустите Antigravity.** Скачайте Google Antigravity, установите и откройте его. Войдите в свой аккаунт Google, когда будет предложено.

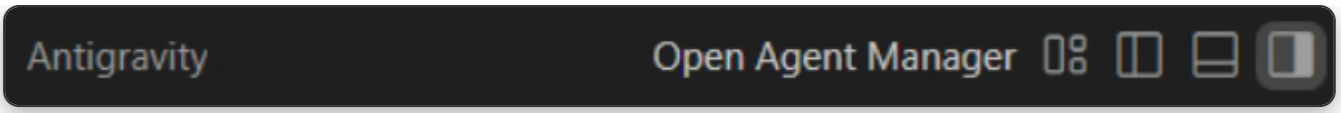
В процессе установки Antigravity попросит вас настроить несколько политик. Для быстрого автономного рабочего процесса мы рекомендуем выбрать **пользовательскую конфигурацию** и установить следующие параметры:

- **Terminal Execution Policy** → Always Proceed
- **Review Policy** → Always Proceed
- **JavaScript Execution** → Always Proceed

С этими настройками агенты Antigravity смогут выполнять команды, записывать файлы и запускать код автономно, не запрашивая подтверждение на каждом шаге. Именно это обеспечивает быстрый и плавный опыт разработки, к которому мы стремимся в этом курсе.

> **Предупреждение о безопасности**: Когда вы разрешаете агентам действовать автономно, и Antigravity, и Claude Code могут выполнять действия в вашей системе без ручного одобрения. Это значит, что нужно быть внимательным к тому, чему вы их подвергаете. Не направляйте агентов на кодовые базы, которые могут содержать вредоносный код, и будьте осторожны со ссылками или ресурсами из ненадежных источников. В эпоху ИИ существуют новые векторы атак — злоумышленники могут создавать контент, специально разработанный для того, чтобы обмануть LLM и заставить их выполнить вредоносные команды. Всегда следите за тем, что делают ваши агенты. Мы рекомендуем настройку «Always Proceed» для скорости, но оставайтесь бдительными и регулярно проверяйте результаты.

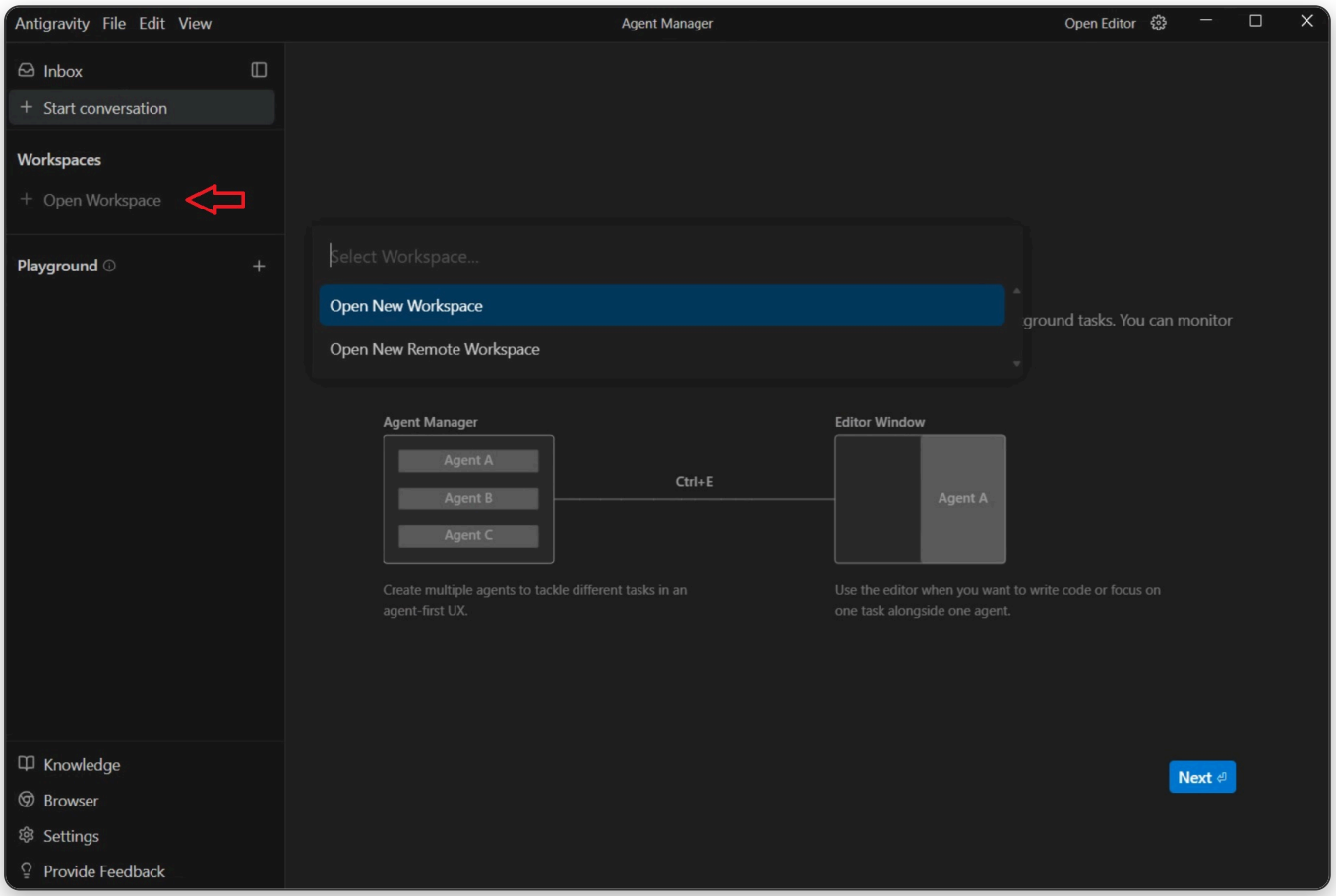
**Шаг 3: Откройте Agent Manager.** Когда Antigravity запущен, нажмите "**Open Agent Manager**" в верхней панели окна.




*Кнопка «Open Agent Manager» в верхней панели Antigravity.*

**Agent Manager** — это центральный узел Antigravity. Здесь вы управляете проектами, начинаете ИИ-разговоры и открываете рабочие пространства для разработки.

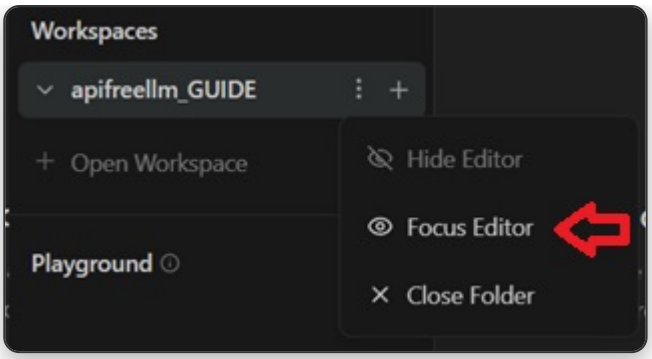
**Шаг 4: Создайте первое рабочее пространство.** В Agent Manager посмотрите на левую боковую панель. В разделе "**Workspaces**" нажмите "**+ Open Workspace**". Появится выпадающее меню — выберите "**Open New Workspace**".




*Нажмите «+ Open Workspace» в левой боковой панели, затем выберите «Open New Workspace».*

Antigravity попросит вас выбрать папку. Перейдите к папке проекта, которую вы создали на шаге 1, и выберите ее. Ваше новое рабочее пространство появится в левой боковой панели в разделе «Workspaces» с именем выбранной папки. Каждое рабочее пространство — это отдельная сессия разработки, по одной на проект. Вы можете создавать столько рабочих пространств, сколько нужно, и переключаться между ними из Agent Manager в любое время.

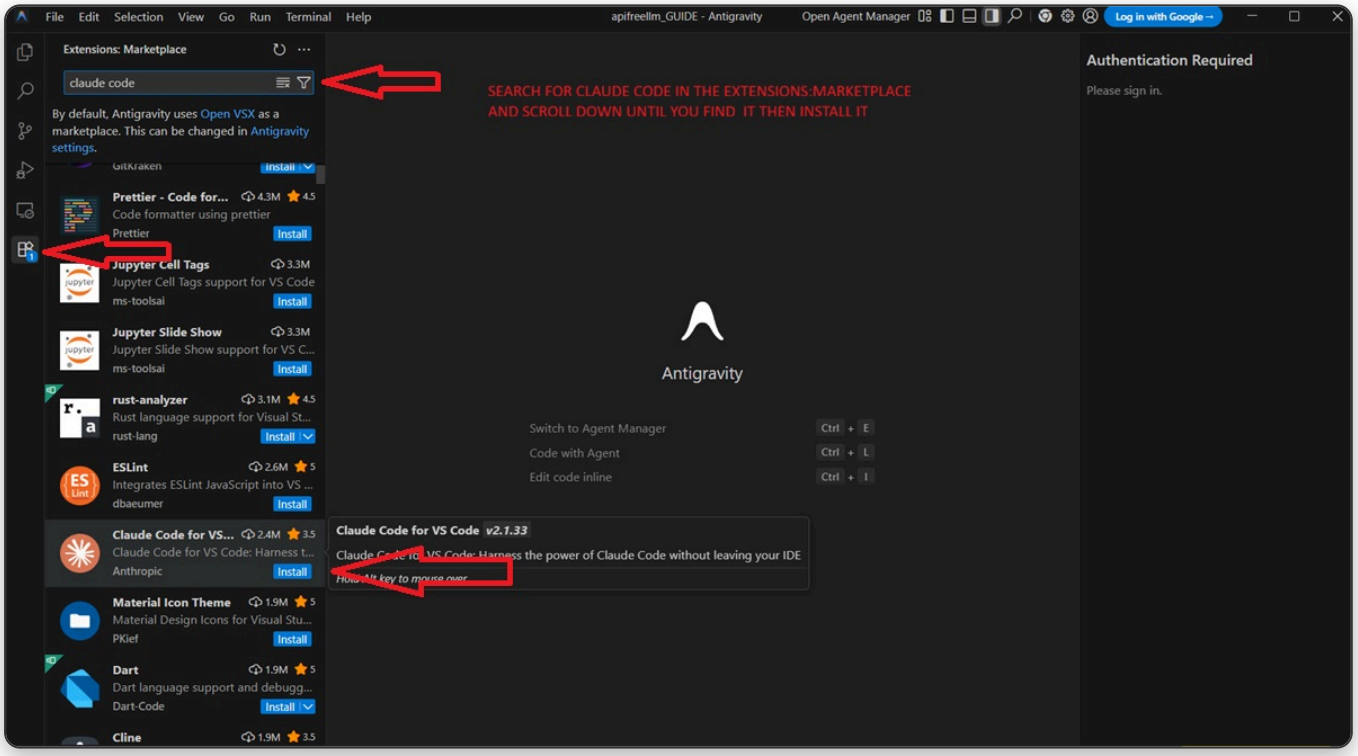
**Шаг 5: Откройте редактор.** Когда рабочее пространство создано, вы увидите его название в боковой панели. Нажмите на **три вертикальные точки** (⋮) рядом с названием рабочего пространства, затем выберите "**Focus Editor**". Это откроет полноценную среду VS Code для этого рабочего пространства, где вы будете писать и редактировать код.




*Нажмите три точки рядом с названием рабочего пространства и выберите «Focus Editor».*

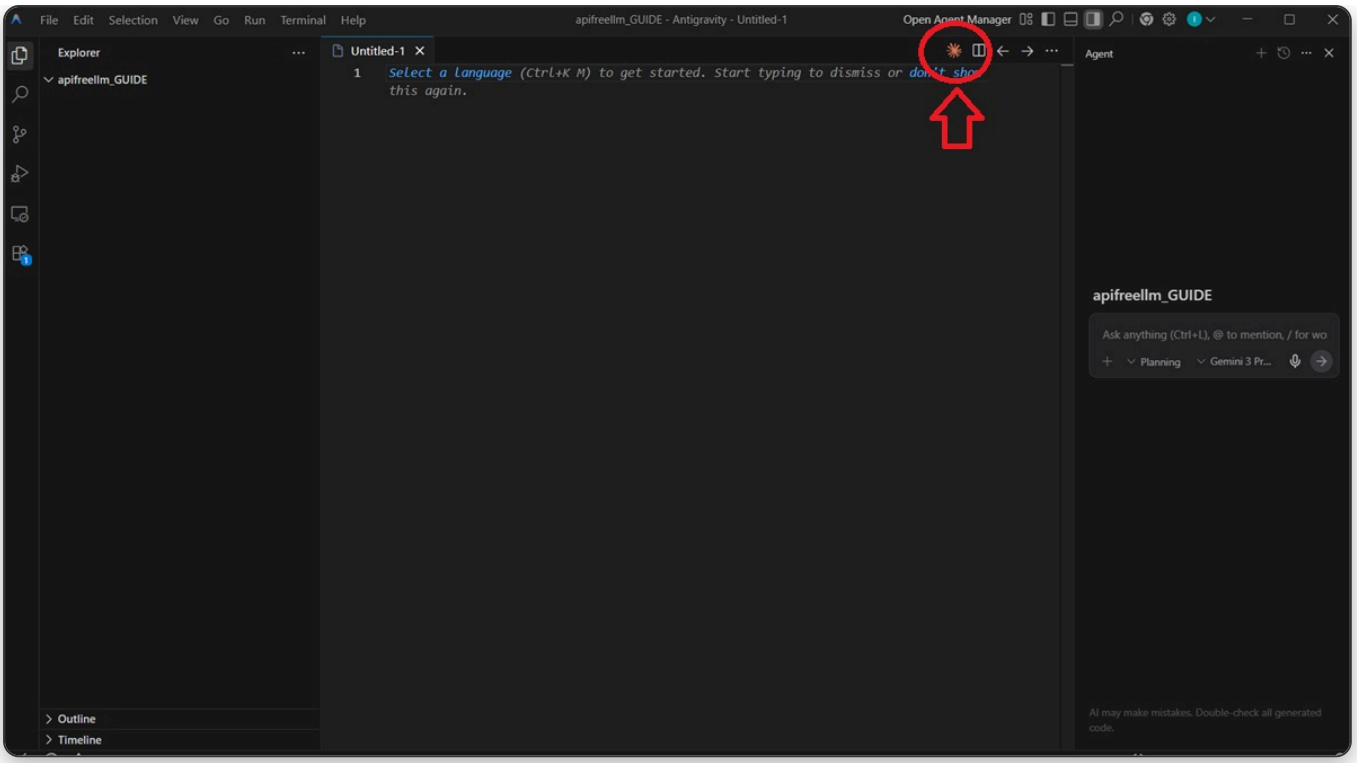
### Установка Claude Code как расширения VS Code

Теперь, когда редактор открыт, пора установить Claude Code. Поскольку редактор основан на VS Code, у вас есть доступ к маркетплейсу расширений. Нажмите на **значок расширений** в левой боковой панели (или нажмите `Ctrl+Shift+X`). В строке поиска введите "**claude code**". Возможно, вам придется прокрутить результаты вниз — ищите "**Claude Code for VS Code**" от Anthropic. Когда найдете, нажмите кнопку **Install**.



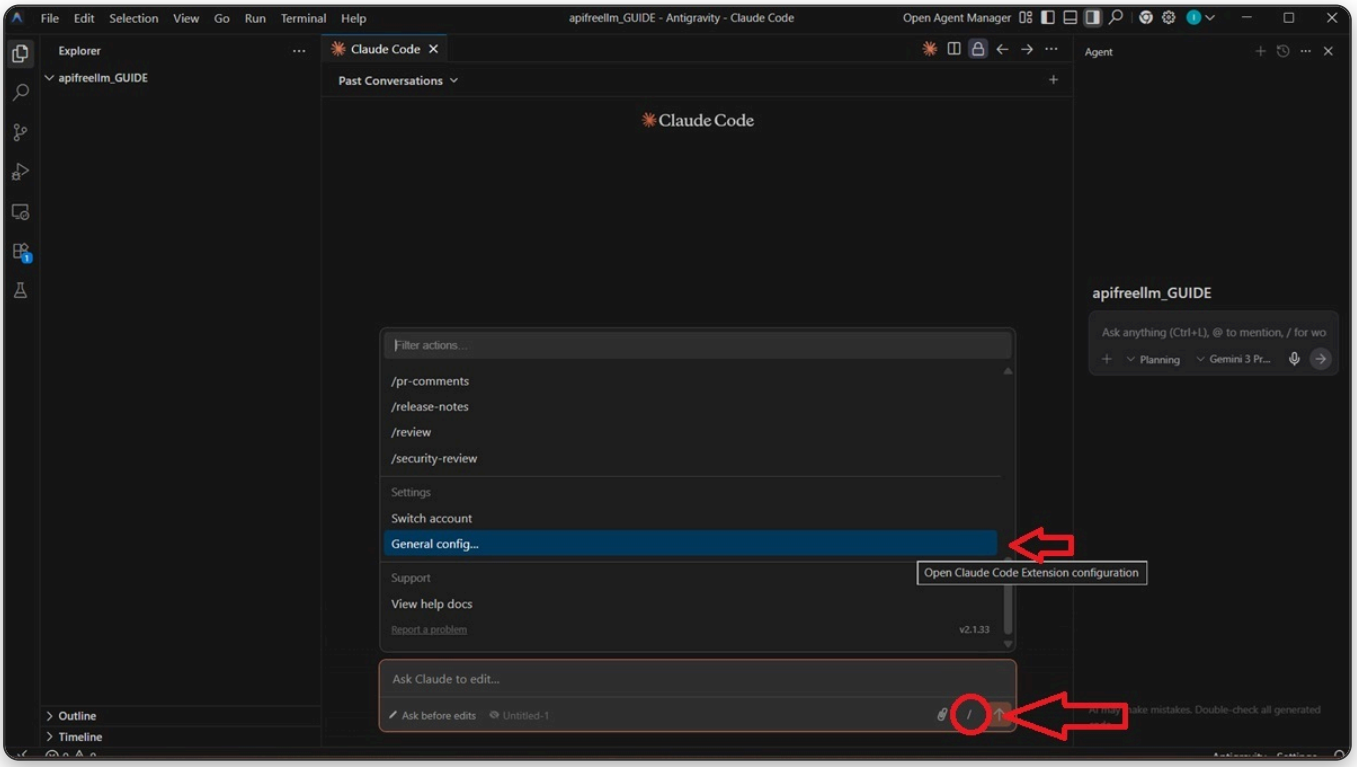

*Найдите «claude code» в маркетплейсе расширений, прокрутите вниз и нажмите Install.*

После установки закройте панель расширений и откройте новый файл (или любой файл в вашем проекте). Вы заметите маленький **значок Claude Code**, появившийся в правой верхней части редактора — он выглядит как маленький оранжевый символ. Нажмите на него, чтобы открыть панель чата Claude Code.



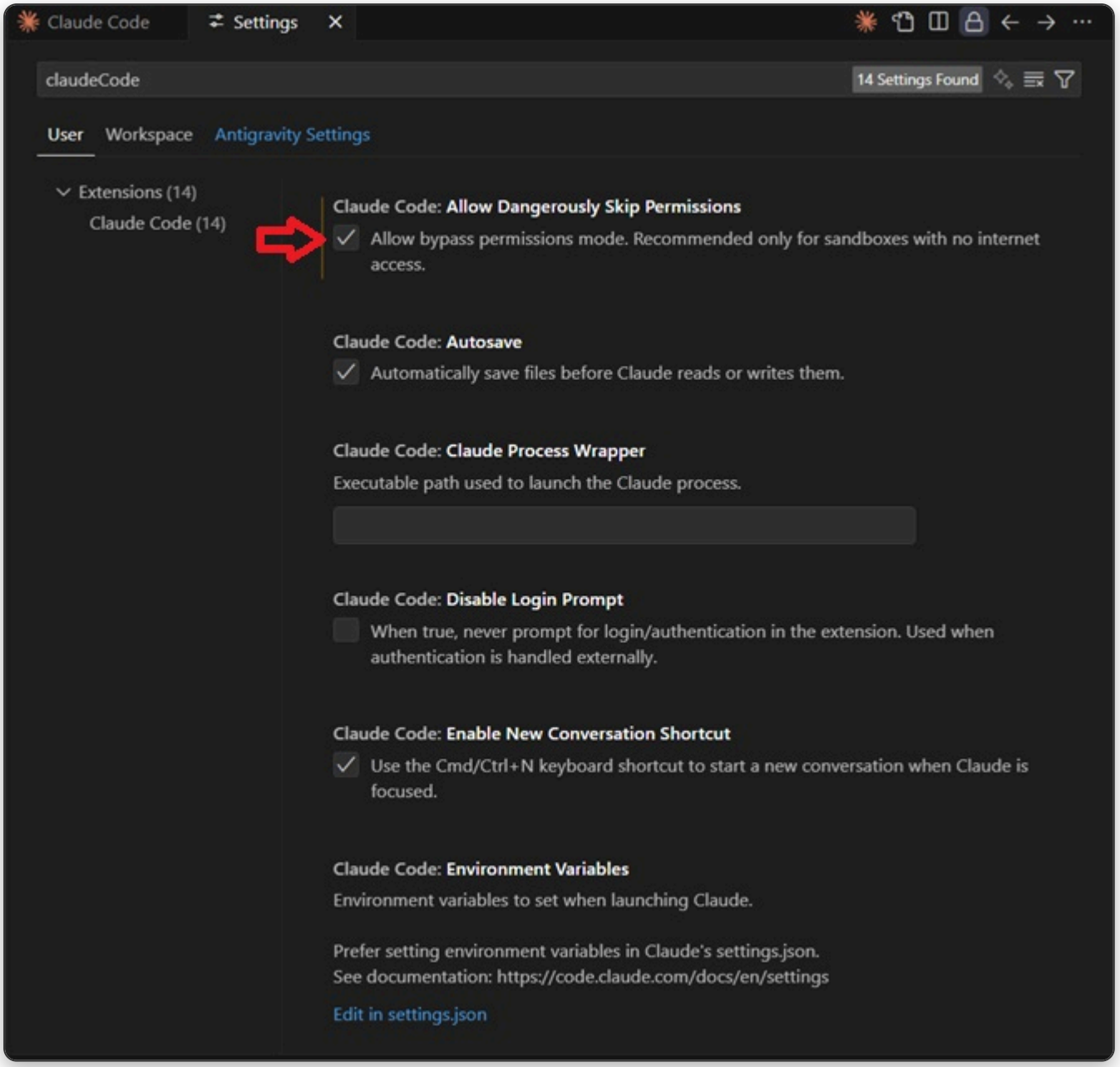

*Кнопка Claude Code (обведена) появляется в правом верхнем углу редактора. Нажмите на нее, чтобы открыть чат Claude Code.*

Claude Code попросит вас войти в ваш аккаунт Anthropic (тот, с подпиской Pro). После входа нужно его настроить. В панели чата Claude Code нажмите кнопку **/** внизу панели. Появится меню с различными командами. Прокрутите вниз до раздела **Settings** и нажмите **"General config..."**, чтобы открыть конфигурацию расширения Claude Code.




*Нажмите кнопку «/» внизу, затем прокрутите до Settings и выберите «General config...», чтобы открыть конфигурацию.*

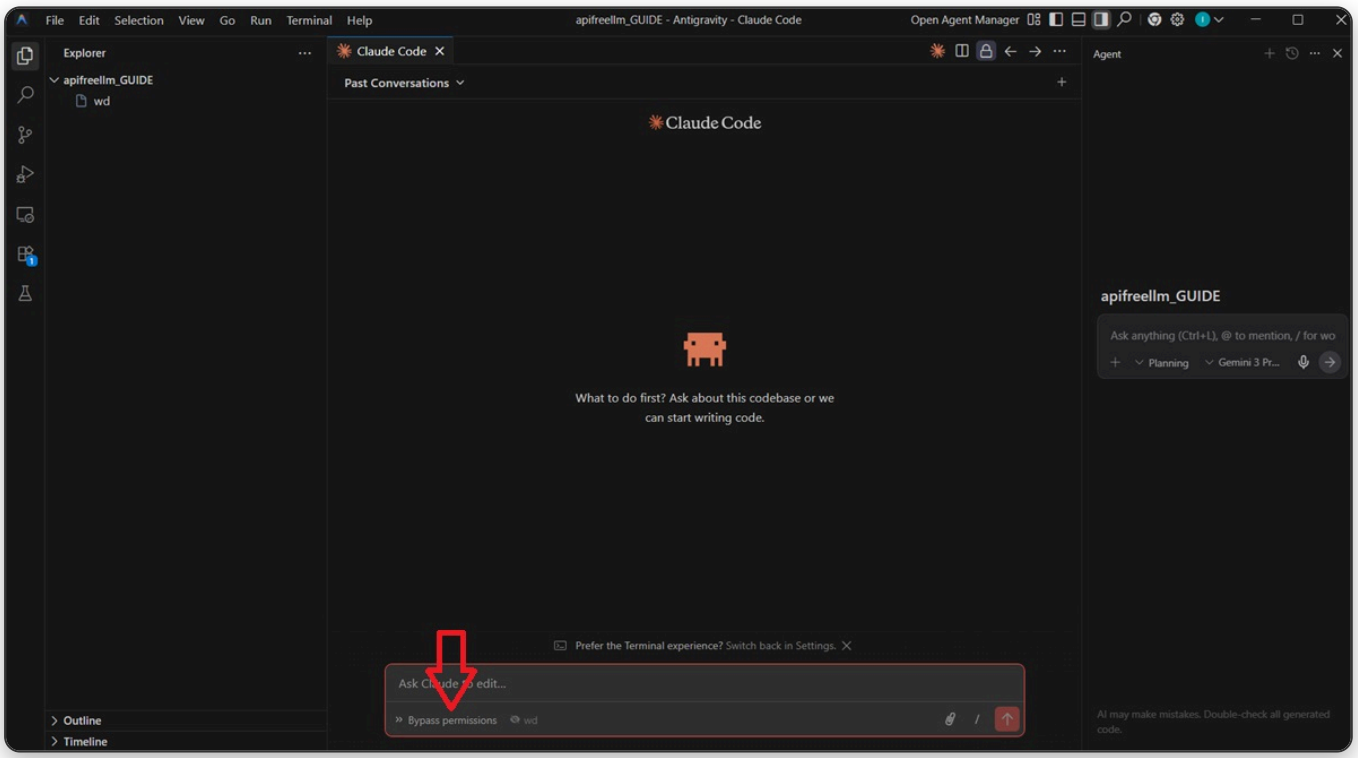
В открывшейся панели конфигурации найдите опцию "Allow Dangerously Skip Permissions". Включите этот переключатель. После активации вы сможете выбрать "Bypass permissions" в качестве режима разрешений, что позволит Claude Code работать полностью автономно — читать файлы, писать код, выполнять команды терминала и вносить изменения без запроса подтверждения на каждом шаге.




*Включите переключатель «Allow Dangerously Skip Permissions» в настройках Claude Code, затем выберите «Bypass permissions».*

> **Важно:** С включенным обходом разрешений Claude Code будет выполнять действия в вашей системе без ручного одобрения. Это необходимо для плавного рабочего процесса разработки, но требует ответственности. Будьте осторожны с кодом, который вы просите его запустить, и никогда не направляйте его на ненадежные репозитории или подозрительные URL-адреса. ИИ-агенты могут быть использованы через инъекцию промптов — вредоносный контент, скрытый в файлах или на сайтах, который обманывает агента и заставляет выполнять вредоносные команды. Всегда проверяйте, что делает Claude Code, особенно при работе с внешними ресурсами.

Мы также рекомендуем включить настройку, которая делает "Bypass permissions" **выбором по умолчанию** для новых чатов, чтобы вам не приходилось выбирать это вручную каждый раз при начале нового разговора. Теперь закройте Claude Code и откройте его снова (нажав кнопку значка Claude Code). С этого момента вы увидите опцию "Bypass permissions", доступную в кнопке выбора разрешений внизу панели чата Claude Code. Нажмите на нее, чтобы активировать.




*Выберите «Bypass permissions» на кнопке, указанной на скриншоте.*

С активным обходом разрешений Claude Code будет выполнять команды, создавать файлы, редактировать код и запускать операции терминала полностью самостоятельно — не останавливаясь для запроса вашего одобрения на каждом шаге. Именно это позволяет вам **автоматизировать 100% рабочего процесса разработки**: вы описываете, что хотите создать, и Claude Code создает это автономно от начала до конца. Больше не нужно нажимать «Accept» на каждое изменение файла или выполнение команды. Вы даете инструкции, а агент делает все остальное.

### Грамотное управление использованием

Одна вещь, которую стоит знать с самого начала: каждое взаимодействие с Claude Code потребляет **токены**, и у вашей подписки есть лимит использования. Хорошая новость в том, что вы можете точно отслеживать, сколько использовали. В панели чата Claude Code нажмите кнопку **/** — среди доступных команд вы найдете **текущее использование**, показывающее, сколько токенов вы потратили и сколько осталось.

Не все модели потребляют токены с одинаковой скоростью. **Claude Opus 4.6** — самая интеллектуальная и способная модель, но она также потребляет больше всего токенов за взаимодействие. Модели меньшего размера, такие как **Sonnet** или **Haiku**, менее мощные, но имеют более высокие лимиты использования и потребляют значительно меньше токенов. Наша рекомендация: используйте **Opus для сложных задач**, требующих глубокого рассуждения, изменений в нескольких файлах или архитектурных решений — именно здесь его интеллект имеет реальное значение. Для более простых задач, таких как быстрые исправления, мелкие правки или простые вопросы, переключайтесь на более легкую модель, чтобы сохранить токены Opus для случаев, когда они действительно важны.

Есть еще одна стратегия экономии токенов Claude: **используйте встроенного агента Antigravity** для вопросов, которые не требуют уровня интеллекта Claude. Antigravity поддерживает несколько моделей, но мы рекомендуем использовать **Gemini** для таких быстрых вопросов — поскольку Antigravity является продуктом Google, Gemini имеет самый высокий лимит использования и также является самой быстрой моделью на платформе. Нужно быстро вспомнить CSS? Хотите узнать синтаксис команды Git? Интересно, как работает библиотека? Спросите Antigravity вместо Claude Code. Так вы сохраните токены Claude для тяжелой работы по разработке, где они дают наибольший эффект.

Как видно на более раннем скриншоте, мы рекомендуем держать **чат Claude Code слева** и **чат агента Antigravity справа**. Такая раскладка бок о бок дает вам мгновенный доступ к обоим агентам в любое время.

> Грамотный рабочий процесс: **Opus для создания, легкие модели для быстрых задач, Antigravity для общих вопросов.** Так вы максимизируете ценность вашей подписки Claude. Если лимит использования Claude заканчивается, помните, что у вас всегда есть агент Gemini от Antigravity рядом для более простых вопросов — используйте его, чтобы сохранить токены Claude для задач разработки, которым они действительно нужны.

### Основная конфигурация

Независимо от способа установки Claude Code, эти шаги конфигурации значительно улучшат ваши результаты:

- **Используйте файл CLAUDE.md** — Разместите файл `CLAUDE.md` в корне вашего проекта. Этот файл автоматически читается Claude Code в начале каждой сессии. Используйте его для описания структуры проекта, соглашений по кодированию, технологического стека и любых правил, которым должен следовать агент. Воспринимайте его как документацию для онбординга вашего ИИ-разработчика.

- **Держите проект организованным** — ИИ-агенты работают значительно лучше с чистыми, хорошо структурированными кодовыми базами. Если ваш код в беспорядке, агент будет выдавать беспорядочный результат. Хорошая структура папок, понятные соглашения об именовании и последовательные паттерны имеют огромное значение.
- **Используйте контроль версий** — Всегда работайте с инициализированным Git. Это дает вам страховку. Если агент допустит ошибку, вы можете мгновенно откатить изменения. Это также позволяет агенту создавать коммиты за вас, что удивительно полезно для отслеживания того, что изменилось и почему.

## Необходимые инструменты: Git и GitHub CLI

Прежде чем мы начнем что-либо создавать, нужно установить два инструмента в вашей системе: **Git** и **GitHub CLI (gh)**. Они фундаментальны для любого современного рабочего процесса разработки и позволяют вашим ИИ-агентам автономно управлять вашими репозиториями кода.

### Почему Git и GitHub CLI важны

**Git** — это система контроля версий, которая отслеживает каждое изменение в вашем проекте. Это ваша страховка: если Claude Code допустит ошибку, вы можете мгновенно откатить. Это также позволяет агенту создавать коммиты, управлять ветками и вести чистую историю эволюции вашего проекта — все автоматически.

**GitHub CLI (gh)** — это инструмент командной строки, который дает прямой доступ к GitHub из терминала. Это ключ к полной автоматизации: как только `gh` установлен и аутентифицирован, Claude Code может создавать репозитории, пушить код, управлять пулл-реквестами, настраивать параметры репозитория, настраивать GitHub Actions для деплоя, добавлять секреты и многое другое — все из своего терминала, без необходимости открывать GitHub в браузере.

### Установка Git и GitHub CLI

Самый простой способ установить эти инструменты — **попросить Claude Code или Antigravity сделать это за вас**. Просто скажите агенту: *«Установи Git и GitHub CLI в моей системе.»* Агент определит вашу операционную систему и выполнит соответствующие команды установки. На Windows он обычно использует `winget` или скачивает установщики; на macOS — `brew`; на Linux — `apt` или пакетный менеджер вашей системы.

Если вы предпочитаете установить их вручную, вы можете скачать Git с официального сайта, а GitHub CLI со страницы релизов на GitHub. Но позволить ИИ сделать это быстрее и позволяет избежать типичных ошибок установки.

### Аутентификация GitHub CLI

После установки вам нужно войти в систему, чтобы `gh` мог получить доступ к вашему аккаунту GitHub. Опять же, вы можете просто попросить агента: *«Войди в GitHub CLI.»* Агент выполнит `gh auth login` и проведет вас через процесс аутентификации, который обычно включает открытие ссылки в браузере и ввод кода. После аутентификации агент имеет полный доступ к вашим репозиториям GitHub.

**Это кардинально меняет игру.** С аутентифицированным `gh` вы можете говорить Claude Code такие вещи, как: *«Создай новый приватный репозиторий my-app, инициализируй проект и запушь код.»* Или позже: *«Настрой GitHub Action, который деплоит на мой сервер при каждом пуше в main.»* Агент справляется со всем — создание файлов, настройка секретов, создание рабочих процессов — без необходимости покидать редактор. Вот как выглядит настоящая автоматизация разработки.

# 3. Ваша первая сборка: от нуля до запуска

> *Ваше окружение готово. Claude Code открыт, Antigravity запущен, Git и GitHub CLI настроены. Пора создать что-то реальное. С этого момента курс переходит от настройки к стратегии — практические техники и инсайты, которые дадут вам реальное преимущество при работе с ИИ-агентами.*

## Никогда не начинайте с нуля

Это самый важный совет во всем курсе: **никогда не создавайте с нуля.**

Хотя Claude Opus — невероятно продвинутая и интеллектуальная модель, она будет допускать ошибки. Любая ИИ-модель их допускает. Она может неправильно понять структуру вашего проекта, использовать устаревшую библиотеку, создать непоследовательную раскладку файлов или сгенерировать код, который не совсем стыкуется, когда проект растет. Начинать с пустой папки означает, что ИИ должен принять сотни решений без опорной точки — и некоторые из этих решений неизбежно будут неверными.

Вот реальность: **99,9% того, что вы хотите создать, уже существует** в какой-то форме. Будь то интернет-магазин, SaaS-панель, портфолио-сайт, приложение для социальных сетей или платформа бронирования — кто-то уже создал что-то подобное. И многие из этих проектов с открытым кодом, доступны как шаблоны на GitHub, готовые к клонированию и настройке.

Ваш рабочий процесс всегда должен начинаться одинаково: **сначала ищите шаблон.** Зайдите на GitHub и поищите проекты с открытым кодом, которые соответствуют тому, что вы хотите создать. Найдите тот, который близок к вашему видению, клонируйте его в папку проекта, а затем направьте на него Claude Code. Теперь вместо создания с нуля агент модифицирует, улучшает и настраивает существующую, работающую кодовую базу. Разница в качестве и скорости огромна.

> **Шаблоны — это не обман, это стратегия.** Профессиональные разработчики постоянно используют бойлерплейты и стартовые наборы. Вы не копируете чей-то продукт. Вы используете структурный фундамент и создаете свой уникальный продукт поверх него. ИИ работает значительно лучше, когда у него есть существующие паттерны для следования, а не когда нужно придумывать все из ничего.

## Выбор правильного стека

Если поиск шаблонов не дал результатов и вам действительно нужно начать новый проект, выбранная технология может сыграть роль — особенно при работе с ИИ-агентами. Тем не менее, единого обязательного выбора не существует. Лучший стек — тот, который быстрее всего приведет вас к работающему продукту.

Claude Code, как и все LLM, фундаментально лучше пишет **веб-код**: HTML, CSS, JavaScript и TypeScript. Это язык интернета, и именно на нем эти модели были обучены больше всего. Чем ближе ваш проект к веб-технологиям, тем лучше будет работать ИИ.

Для **веб-приложений Next.js** — отличный выбор, если вы можете его найти. Это современный React-фреймворк со встроенным серверным рендерингом, что критически важно для SEO. Claude Code исключительно хорошо работает с проектами на Next.js: он понимает маршрутизацию на основе файлов, API-маршруты, серверные компоненты и всю экосистему. Вы найдете огромное количество шаблонов на Next.js на GitHub для практически любого типа приложения.

Для **десктопных приложений** (исполняемые файлы для Windows, macOS или Linux) **Electron** — отличный вариант. Electron позволяет создавать десктопные приложения с использованием HTML, CSS и JavaScript — тех же веб-технологий, в которых Claude превосходен. Поскольку интерфейс — это, по сути, веб-страница, отрендеренная внутри нативного окна, ИИ может создавать красивые, функциональные десктопные приложения с той же легкостью, что и сайты.

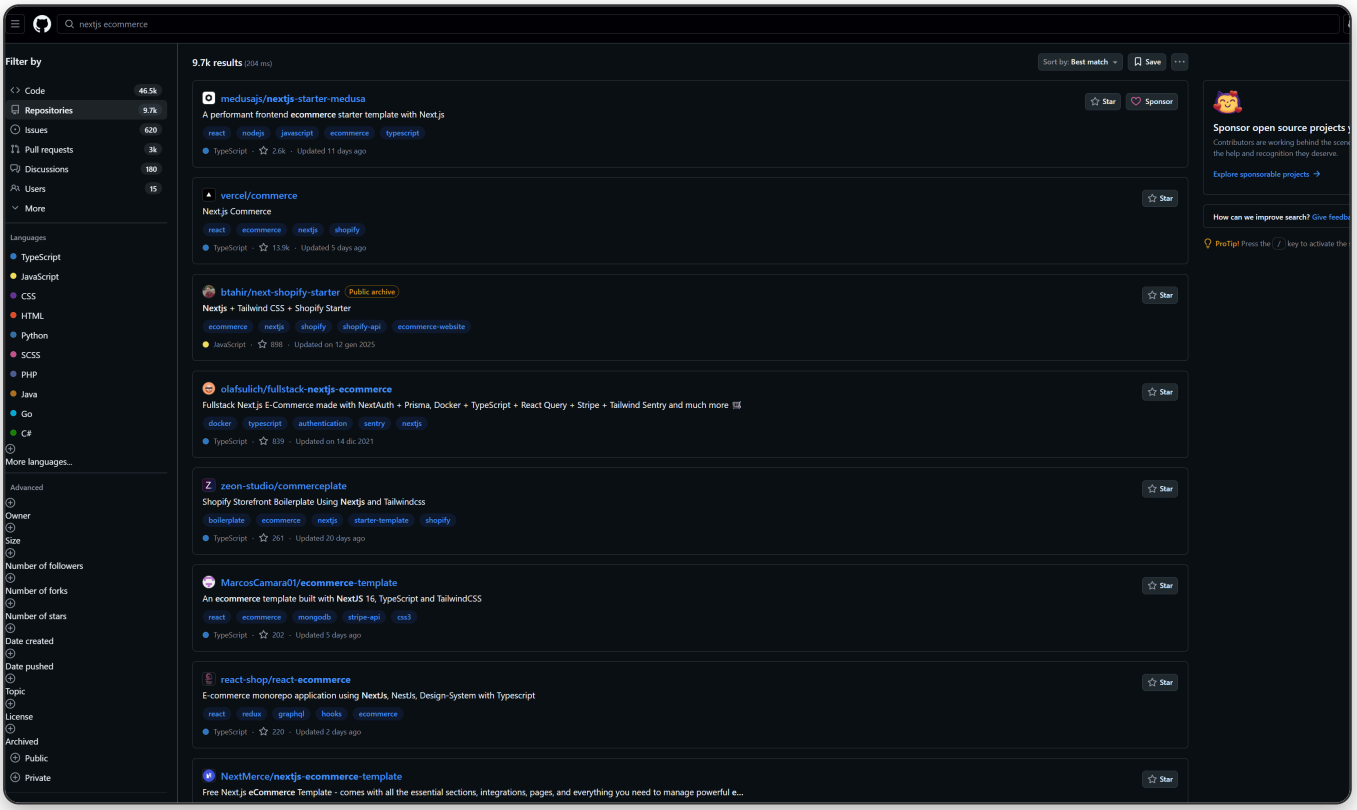
Но вот важный нюанс: **хорошо сделанный шаблон на старом стеке почти всегда лучше, чем начало с нуля на современном.** Вы можете найти проект на PHP, jQuery или Laravel, который точно соответствует вашим потребностям — полный, хорошо структурированный, проверенный временем, со всеми уже реализованными функциями. В таком случае используйте его. ИИ-агенты могут работать с любой технологией, и время, которое вы сэкономите, имея надежный, готовый фундамент, с лихвой перевешивает теоретические преимущества более нового фреймворка. Claude Code отлично понимает и модифицирует PHP, Python, Ruby или любую другую кодовую базу.

Приоритет прост: **найдите лучший доступный шаблон.** Если вы нашли два шаблона схожего качества и один использует Next.js, а другой — PHP, выбирайте Next.js. Но если шаблон на PHP более полный, функциональный и поддерживаемый — берите его без колебаний. Качество и полнота отправной точки важнее, чем современность стека.

> **Правило:** используйте то, что приближает вас к цели. Стремитесь к современным веб-технологиям (Next.js, Electron, React Native), когда они доступны, но никогда не отвергайте отличный шаблон только потому, что он использует старый стек. Время, сэкономленное благодаря надежному фундаменту, всегда ценнее, чем чистота стека.

### Практический пример: поиск шаблона

Допустим, вы хотите создать интернет-магазин. Вместо того чтобы говорить Claude Code *«Создай мне интернет-магазин с нуля»*, откройте GitHub и поищите что-то вроде **"nextjs ecommerce"**. Вы найдете десятки готовых проектов со списками товаров, корзинами покупок, процессами оформления заказов и интеграцией платежей.




*Быстрый поиск на GitHub по запросу «nextjs ecommerce» уже показывает несколько перспективных шаблонов.*

Одна важная вещь, о которой стоит помнить: многие шаблоны на GitHub — это **фримиум-проекты** — ядро открыто и бесплатно, но некоторые функции или премиум-версии требуют оплаты. Всегда проверяйте README и лицензию репозитория, прежде чем выбрать шаблон. Избегайте всего, что требует платных подписок или скрытых расходов, которые вам не нужны.

Глядя на результаты поиска, мы можем заметить **fullstack-nextjs-ecommerce** — и это отличная отправная точка. Давайте разберем почему. Проект использует Next.js с TypeScript, что именно то, что нам нужно для разработки с ИИ. Но что действительно выделяется — это то, что уже интегрировано: **Stripe для платежей**, **PostgreSQL с Prisma** в качестве базы данных и **NextAuth для аутентификации**. Это три наиболее критических — и наиболее подверженных ошибкам — части любого веб-приложения.

Наличие **уже интегрированного Stripe** особенно ценно. Обработка платежей включает вебхуки, управление сессиями, обработку ошибок и пограничные случаи, которые на удивление сложно реализовать правильно. Когда что-то идет не так с платежами, страдают ваши клиенты — неудавшиеся списания, дублирование платежей или сломанные процессы оплаты — это те баги, которые разрушают доверие и стоят вам реальных денег. Начать с шаблона, в котором уже есть работающая интеграция Stripe, избавляет от этих проблем.

Проект также использует **PostgreSQL** в качестве базы данных, что является надежным выбором. Postgres надежен, хорошо документирован, предлагает чуть лучшие настройки безопасности по сравнению с альтернативами вроде MySQL, и может быть легко размещен на Amazon AWS, Hetzner или аналогичных провайдерах — мы рассмотрим настройку сервера и деплой в следующей главе. Тот факт, что **NextAuth** уже подключен, означает, что аутентификация пользователей работает из коробки — еще одна сложная часть, которую не нужно создавать с нуля.

В этом сила начала с правильного шаблона: вместо того чтобы тратить дни (или недели) на настройку платежей, базы данных и аутентификации — все вещи, которые легко сделать неправильно — вы начинаете с проекта, где эти критические системы уже работают. Вы можете полностью сосредоточиться на настройке продукта под свое видение: изменение дизайна, добавление товаров, модификация бизнес-логики и создание функций, которые делают ваш магазин уникальным.

Когда вы нашли шаблон, следующий шаг прост: **скачайте его и поместите в папку рабочего пространства.** Откройте эту папку с помощью Claude Code (или Antigravity) и начните работу. Вы можете попросить Claude Code модифицировать шаблон напрямую — изменить брендинг, добавить новые страницы, переработать макет, интегрировать новые функции — или использовать шаблон как **справочник** для собственного проекта. Даже если вы создаете что-то немного другое, наличие шаблона в том же рабочем пространстве означает, что Claude Code может изучить его, исследовать паттерны кода и использовать их как основу для того, что он пишет.

Это ключевой момент: **всегда давайте Claude Code что-то для справки.** Когда у агента есть надежная, работающая кодовая база для изучения, он пишет значительно лучший код. Он следует тем же паттернам, использует те же соглашения и выдает последовательный и надежный результат. Когда ему нечего использовать как справку и приходится генерировать все с нуля, именно тогда возникают ошибки — непоследовательная структура файлов, неправильные версии библиотек, код, который не стыкуется. Шаблон действует как якорь, который удерживает ИИ и обеспечивает высококачественные, связные результаты.

> **Что если в шаблоне нет платежей или базы данных?** Это тоже нормально. Этот курс включает готовые файлы интеграции Stripe, которые вы можете передать напрямую Claude Code для интеграции платежей в любой проект. Для базы данных мы рекомендуем PostgreSQL или ту базу данных, которую уже использует шаблон — не боритесь с выбором шаблона, если для этого нет веской причины. То же касается аутентификации: если шаблон использует NextAuth, Clerk или любую другую систему авторизации, работайте с ней. Цель — использовать то, что уже создано, а не заменять все.

# 4. От кода до продакшена

> *Ваш продукт создан. Теперь ему нужно принимать платежи, работать на сервере и быть быстрым и безопасным для пользователей по всему миру. Эта глава охватывает основные сервисы, которые превращают ваш проект из локального кода в работающий, готовый к продакшену бизнес.*

> **Замечание об этой главе.** Мы стараемся быть краткими и по делу. Наша цель — рассказать вам **что** нужно сделать и **почему** — а не писать длинные туториалы, которые отнимают ваше время. Любая LLM (Claude, Gemini, ChatGPT) может объяснить детали, провести вас через каждый шаг и ответить на ваши вопросы гораздо лучше, чем статичная страница. Когда что-то непонятно, просто спросите агента: *«Я прохожу курс, и там сказано, что мне нужно [сделать X]. Можешь объяснить, что это значит, и провести меня через это?»* Это быстрее, более персонализировано и всегда актуально.

## Платежи через Stripe

Если ваш продукт что-то продает, вам нужен платежный процессор. **Stripe** — это индустриальный стандарт: надежный, хорошо документированный и поддерживается практически каждым ИИ-агентом из-за его повсеместного использования.

Вот что вам нужно сделать:

- **Создайте аккаунт Stripe** на stripe.com. Вы получите **API-ключи** (публичный ключ для фронтенда, секретный ключ для бэкенда) и доступ как к **тестовому режиму** (фиктивные платежи для разработки), так и к **боевому режиму** (реальные деньги).
- **Настройте вебхуки** в панели управления Stripe. Вебхуки — это URL-адреса на вашем сервере, которые Stripe вызывает, когда что-то происходит — платеж успешен, подписка продлена, списание не удалось. Так ваше приложение узнает, когда активировать подписку, подтвердить заказ или обработать ошибку. Вам нужны вебхуки как для **локального тестирования** (у Stripe есть CLI-инструмент, который пересылает события на ваш localhost), так и для **продакшена** (указывая на ваш боевой сервер). Всегда отлаживайте все локально, прежде чем выходить в продакшен.
- **Интегрируйте Stripe в ваш проект.** Если в вашем шаблоне уже есть Stripe, просто подставьте свои API-ключи. Если нет, этот курс включает готовые файлы интеграции Stripe — поместите их в рабочее пространство и попросите Claude Code их интегрировать. Stripe поддерживает разовые платежи и повторяющиеся подписки, оба через хостируемые страницы оплаты, которые берут на себя валидацию карты, 3D Secure и PCI-совместимость.

Одно важное правило: **всегда проверяйте платежи на сервере через вебхуки**, никогда не доверяйте фронтенду. Вебхук — это когда Stripe напрямую сообщает вашему серверу, что деньги действительно были переведены — это единственный надежный источник истины.

## Хостинг: AWS, Hetzner и другие

Вашему приложению нужно где-то жить. Хорошая новость: **AWS дает вам бесплатный уровень** при регистрации — на **целый год** вы получаете **сервер t2.micro** и **микро-базу данных** полностью бесплатно. Этого достаточно для хостинга вашего первого проекта, пока вы проверяете идею и набираете пользователей.

Не знаете, как создать аккаунт AWS или использовать бесплатный уровень? Просто спросите Claude или Antigravity — они проведут вас через каждый шаг.

Когда вы перерастете бесплатный уровень, вот что нужно знать о размерах серверов:

- **t3.small** — оптимальный выбор для большинства приложений — хороший процессор, достаточно оперативной памяти и разумная цена (~$16/месяц на AWS). Буква «t» обозначает поколение инстансов; старые поколения (t2 и др.) иногда могут стоить дороже при меньшей производительности, поэтому выбирайте самое последнее доступное.
- **Hetzner** — европейская альтернатива, которая *значительно* дешевле — вы можете получить производительность, сопоставимую с t3.small, примерно за **$4/месяц**. Это примерно в 4 раза дешевле AWS при схожих характеристиках.
- **Не каждому приложению нужен сервер.** Если ваш проект — статический сайт или JAMstack-приложение, вам, возможно, вообще не нужен выделенный сервер. Платформы вроде Vercel, Netlify или даже Cloudflare Pages могут хостить его бесплатно или почти бесплатно. Всегда спрашивайте вашу LLM: *«Какой лучший хостинг для моего конкретного приложения?»*

Наша рекомендация: **начните с бесплатного уровня AWS** на первый год, затем оцените, что больше подходит — Hetzner или другой провайдер — с учетом вашего бюджета и расположения аудитории. Если большинство ваших пользователей в Европе, немецкие серверы Hetzner обеспечат меньшую задержку. Если ваша аудитория глобальная или находится в США, регионы AWS могут подойти лучше.

> **SSH-ключи и управление сервером через ИИ.** Чтобы позволить Claude Code подключаться к вашему серверу и управлять им удаленно, вам нужно настроить **SSH-ключ**. Попросите Claude сгенерировать его и настроить на вашем сервере. После подключения Claude может деплоить код, управлять сервисами, устранять неполадки — все из вашего терминала. Для задач управления сервером мы рекомендуем использовать **Opus** для лучших результатов, хотя Sonnet тоже подойдет, если хотите сэкономить токены (с чуть большей вероятностью ошибок).

## Cloudflare: производительность и защита

Когда ваше приложение на сервере, вам нужно что-то, стоящее перед ним, чтобы защитить и ускорить его. Это **Cloudflare**. Хорошая новость: **бесплатного плана более чем достаточно** для подавляющего большинства сайтов. Платный план вам не нужен.

Но сначала вам понадобится **доменное имя**. Мы рекомендуем покупать его напрямую у **Cloudflare** или **Namecheap** — оба надежны и предлагают адекватные цены. Когда у вас есть домен, нужно настроить **DNS**, чтобы он указывал на IP-адрес вашего сервера AWS или Hetzner. Просто спросите вашу LLM: *«Я купил домен на [Cloudflare/Namecheap]. Как настроить DNS, чтобы он указывал на мой сервер с адресом [ваш IP]?»* Она проведет вас через это.

Почему Cloudflare так важен? Он защищает ваш сайт от **DDoS-атак**, различных **уязвимостей безопасности** и распространенных веб-угроз. Он также предоставляет **глобальный кэш**, то есть ваш контент раздается с серверов, расположенных близко к вашим пользователям, что делает сайт быстрее по всему миру. Без сервиса вроде Cloudflare вы подвергаете свой сайт серьезным рискам — особенно сейчас, в эпоху ИИ, когда любой может использовать ИИ-инструменты для поиска уязвимостей, эксплуатации ошибок конфигурации или даже кражи данных с плохо защищенных сайтов. Не пропускайте это.

> **Быстрый совет: режим Flexible SSL.** При настройке Cloudflare вы можете выбрать режим **Flexible** SSL. Это означает, что соединение между Cloudflare и вашим сервером использует обычный HTTP, но соединение между Cloudflare и конечными пользователями — HTTPS, поэтому посетители видят значок защищенного соединения. Это избавляет вас от необходимости настраивать SSL-сертификаты на сервере, что отлично для быстрого старта. Для продакшен-приложений, обрабатывающих конфиденциальные данные, вам со временем стоит перейти на режим **Full** (шифрование от начала до конца). Но для первых тестов и запусков Flexible вполне подходит и экономит много времени на настройку. Спросите вашу LLM, чтобы она объяснила различия, если не уверены, какой режим подходит вашему проекту.

Cloudflare предлагает гораздо больше, чем просто защиту. Бесплатный план включает мощные функции, о которых многие разработчики даже не знают:

- **Cloudflare Pages** — бесплатный статический хостинг для фронтенд-приложений (React, Next.js static export, Vue и т.д.). Вы пушите на GitHub, Cloudflare собирает и деплоит автоматически. Нулевая конфигурация, нулевые затраты. Идеально для лендингов, портфолио и JAMstack-приложений.
- **Edge Functions (Workers)** — запускайте серверлесс-код на границе сети, близко к вашим пользователям, на бесплатном плане. Отлично для API-маршрутов, редиректов, A/B-тестирования и лёгкой бэкенд-логики без выделенного сервера.
- **CDN & Caching** — ваши статические ресурсы (изображения, CSS, JS) кэшируются глобально, делая ваш сайт молниеносно быстрым из любой точки мира.

Ключевой вопрос: **действительно ли вашему приложению нужен выделенный сервер, или Cloudflare может справиться бесплатно?** Спросите Claude: *"Я создаю [опишите ваше*

*приложение]. Мне нужен выделенный сервер на AWS/Hetzner, или я могу использовать Cloudflare Pages и Workers бесплатно?"* Claude проанализирует ваш конкретный случай и подскажет лучший вариант. Вы можете удивиться, сколько проектов могут полностью работать на бесплатном тарифе Cloudflare.

## API-ключи: Автоматизируйте всё

Вот совет, который меняет правила игры и который большинство туториалов пропускают: **создайте API-ключи для ваших хостинг-провайдеров** и передайте их Claude. Это позволит Claude управлять вашей инфраструктурой прямо из терминала — без кликов по дашбордам, без копирования и вставки, без ручной работы.

- **API-ключ Cloudflare** — перейдите в дашборд Cloudflare → My Profile → API Tokens → создайте токен. С ним Claude может автоматически настраивать DNS-записи, создавать проекты Pages, управлять Workers, обновлять настройки SSL и многое другое. Скажите Claude: *"Вот мой API-токен Cloudflare. Настрой DNS для моего домена, указывающий на IP моего сервера."* Готово за секунды.
- **AWS Access Keys** — перейдите в AWS IAM → создайте access key. С ним Claude может управлять EC2-инстансами, настраивать security groups, создавать базы данных RDS, управлять S3-бакетами и программно управлять всей вашей AWS-инфраструктурой. Скажите Claude: *"Вот мои AWS-учётные данные. Запусти EC2-инстанс t3.small в us-east-1 и настрой security groups для веб-приложения."*
- **Hetzner API Token** — перейдите в вашу Hetzner Cloud Console → проект → Security → API Tokens → сгенерируйте токен. Claude сможет создавать серверы, настраивать файрволы, управлять снапшотами и управлять всей вашей инфраструктурой Hetzner. Скажите Claude: *"Вот мой API-токен Hetzner. Создай сервер CX22 в Нюрнберге с Ubuntu."*

> **Замечание по безопасности API-ключей.** Эти API-ключи мощные — обращайтесь с ними как с паролями. **Никогда не коммитьте их в Git**, не делитесь ими публично и не вставляйте в чат-интерфейсы, которым не доверяете. Храните их в файле `.env` или передавайте напрямую Claude в вашей терминальной сессии. Если ключ скомпрометирован, немедленно отзовите его в дашборде провайдера и сгенерируйте новый. Также при создании API-токенов **всегда используйте принцип наименьших привилегий**: давайте только те разрешения, которые действительно нужны, а не полный административный доступ.

Комбинация этих API-ключей с Claude невероятно мощная. Вы можете буквально сказать: *"У меня Next.js-приложение. Задеплой его на Cloudflare Pages, настрой пользовательский домен и сконфигурируй DNS — вот мои API-ключи."* И Claude сделает всё автоматически. Или: *"Создай EC2-инстанс на AWS, установи Node.js и PM2, настрой nginx, настрой Cloudflare DNS и задеплой моё приложение."* Полная настройка инфраструктуры за минуты, а не часы.

## CI/CD с GitHub Actions

Помните, когда мы настраивали Git и GitHub CLI в главе 2? Вот где это все окупается. С аутентифицированным `gh` вы можете попросить Claude Code **запушить проект в приватный репозиторий GitHub**, настроить **GitHub Actions**, сконфигурировать все необходимые **секреты** (SSH-ключ вашего сервера, переменные окружения и т.д.) и создать автоматический пайплайн деплоя — все из терминала, не заходя в интерфейс GitHub.

Идея проста: когда GitHub Actions настроен, **каждый раз, когда Claude Code пушит код в ваш репозиторий, он автоматически деплоится на ваш сервер.** Никакого ручного SSH, копирования файлов, запуска команд на сервере. Claude пушит, GitHub Actions подхватывает, подключается к вашему серверу AWS или Hetzner по SSH и все деплоит. Полностью автоматически.

Просто скажите Claude: *«Запушь этот проект в новый приватный репозиторий на GitHub, настрой GitHub Action, который деплоит на мой сервер при каждом пуше в main. Вот IP моего сервера и SSH-ключ.»* Claude уже знает, как это делать — он создаст файл рабочего процесса, добавит SSH-ключ и IP сервера как секреты GitHub и настроит весь пайплайн. Именно для этого мы установили GitHub CLI ранее.

> **Ловушка с динамическим IP.** Статические IP обычно стоят дополнительных денег и на AWS, и на Hetzner, поэтому вы, скорее всего, будете использовать **динамический IP** для экономии. Это значит, что каждый раз, когда вы останавливаете и перезапускаете сервер, IP меняется. Когда это происходит, вам нужно обновить его в **двух местах**: DNS-запись в Cloudflare и секрет `SERVER_IP` в GitHub. Попросите Claude хранить IP сервера как секрет `SERVER_IP` в GitHub Actions, чтобы при изменении вам нужно было обновить только одну переменную. Также попросите Claude напоминать вам проверить и обновить IP, если деплой не удался — изменившийся IP почти всегда является причиной.

# 5. Маркетинг и SEO-тактики

> *Ваш продукт запущен. Теперь мир должен узнать о его существовании. Наиболее эффективный маркетинг для инди-продуктов — органический: бесплатный, креативный и удивительно мощный, если все сделать правильно. Эта глава охватывает именно те тактики, которые мы используем.*

## Стратегии органического роста

Давайте будем честны: **лучший маркетинг не выглядит как маркетинг.** Интернет перенасыщен рекламой, и у людей выработался рефлекс игнорировать все, что выглядит как продвижение. Что действительно работает — это **скрытый маркетинг** — контент, который выглядит как искренняя

информация, вопрос или рекомендация, а не реклама. Люди от природы любопытны и любят помогать своими советами. Используйте это.

**Reddit** — одна из самых мощных платформ для этого. Он огромный, отлично индексируется Google и полон нишевых сообществ, где уже обитает ваша целевая аудитория. Но вот ключевой момент: **никогда не постите агрессивную рекламу.** Не пишите *«Посмотрите мой крутой новый сайт!»* — это получит минусы, будет удалено или забанено. Вместо этого напишите что-то вроде:

- *«Кто-нибудь может посоветовать сайты, похожие на [известный конкурент] или [ваш сайт]? Ищу альтернативы.»*
- *«Кто-нибудь пробовал [категория вашего продукта]? Нашел [ваш сайт], но интересно, какие еще есть варианты.»*
- Отвечайте на вопросы в релевантных сабреддитах и естественно упоминайте свой продукт там, где он действительно помогает.

Именно так работает большинство успешного инди-маркетинга на Reddit. Вы не лжете — вы подаете свой продукт как находку в рамках искренней беседы. Люди кликают, потому что им любопытно, а не потому что чувствуют давление продаж. И вот бонус: **посты на Reddit индексируются Google**, поэтому удачно размещенный тред может приносить вам органический поисковый трафик месяцами и даже годами.

Вы также можете **продвинуть пост на Reddit** за небольшой бюджет для временного буста. Это поднимает его выше в результатах поиска, позволяет Google быстрее его проиндексировать и дает начальную видимость. Спросите вашу LLM, как работает продвижение постов на Reddit и какой бюджет имеет смысл.

> **Трюк с воронкой.** На главной странице или лендинге разместите что-то, что **привлекает людей независимо от вашего основного продукта** — бесплатный инструмент, актуальную тему, полезную информацию или что-то сейчас популярное. Это работает как **воронка**: люди приходят за бесплатным/интересным контентом, обнаруживают ваш продукт, и определенный процент из них конвертируется. Воспринимайте это как приманку, которая дает реальную ценность и одновременно ведет к тому, что вы продаете.

## SEO, контент и дистрибуция

Прежде чем начинать любой маркетинг, настройте **аналитику и отслеживание**. Вам нужны две вещи:

- **Google Analytics** — добавьте код отслеживания на ваш сайт (попросите Claude интегрировать). Есть также **мобильное приложение**, чтобы проверять трафик с телефона в любое время. Это показывает общее количество посещений, поведение пользователей, источники трафика и данные о конверсиях.
- **Google Search Console** — настройте это и подключите к Google Analytics. Search Console специально отслеживает ваш **органический трафик из Google**: какие поисковые запросы

приводят людей на ваш сайт, как часто вы появляетесь в результатах поиска и ваш показатель кликабельности. Спросите вашу LLM, как настроить и подключить оба инструмента.

Если у вас есть бюджет, **Google Ads** может дать начальный толчок. Запуск рекламы на короткое время помогает завоевать доверие алгоритма Google и привлекает ранний трафик, пока растет ваше органическое SEO. Но расходы быстро накапливаются, поэтому относитесь к этому как к краткосрочному ускорителю, а не долгосрочной стратегии. Ваша LLM может подробно объяснить настройку Google Ads и бюджетирование.

Что касается самого SEO, начните с основ. Откройте **DevTools** в браузере (F12), перейдите в **Lighthouse** и сгенерируйте отчет. Это даст вам оценки по производительности, доступности, лучшим практикам и **SEO**. Исправьте то, что он укажет — и да, вы можете попросить Claude Code заняться исправлениями.

Ключевые действия по SEO:

- **Добавьте переводы** на ваши страницы. Многоязычная поддержка значительно увеличивает ваш охват. Попросите Claude Code настроить интернационализацию для вашего проекта.
- **Добавьте правильные мета-теги** — title, description, теги Open Graph для расшаривания в соцсетях, структурированные данные. Они напрямую влияют на то, как ваш сайт отображается в результатах поиска Google.
- **Создайте и отправьте карту сайта.** Сгенерируйте актуальную карту сайта и загрузите ее в Google Search Console. Это сообщает Google, какие именно страницы есть на вашем сайте, и помогает им быстрее проиндексироваться.

> **SEO требует терпения.** Не ждите органический трафик за одну ночь — требуются недели или месяцы, чтобы Google правильно проиндексировал и ранжировал ваши страницы. Но когда это заработает, это **бесплатный трафик, который продолжает приходить**, без каких-либо затрат с вашей стороны. Клики, за которые вы бы иначе платили сотни евро через Google Ads, будут приходить бесплатно через органический поиск. А пока работайте на социальных платформах вроде Reddit, чтобы создать начальную видимость и обратные ссылки. SEO — это игра вдолгую, но это самая ценная маркетинговая инвестиция, которую вы можете сделать.

### Зарождающийся тренд: трафик от ИИ

Есть новый и быстро растущий источник трафика, на который большинство людей пока не обращает внимания: **LLM, рекомендующие ваш сайт.** ChatGPT, Gemini, Claude и другие ИИ-ассистенты все чаще используются как поисковые системы. Когда кто-то спрашивает *«Какой хороший сайт для [категория вашего продукта]?»*, эти модели могут и действительно рекомендуют конкретные сайты — и это приносит реальный трафик. Значительная часть наших собственных посещений приходит от ChatGPT и других LLM.

Чтобы воспользоваться этим, зайдите в **панель управления Cloudflare** и убедитесь, что вы **отключили опцию, блокирующую ИИ-краулеры**. Многие сайты по умолчанию блокируют ИИ-ботов, что не позволяет LLM узнать о вашем контенте. Разрешив ИИ-краулерам доступ к вашему

сайту, вы позволяете этим моделям индексировать ваши страницы, понимать, что вы предлагаете, и потенциально рекомендовать вас пользователям в будущем. По сути, это **бесплатное SEO для эры ИИ** — и эффект может быть огромным.

> **Думайте шире, чем Google.** Традиционное SEO нацелено на поиск Google. Но рекомендации от ИИ становятся крупным источником трафика — и этот тренд только ускоряется. Убедитесь, что ваш сайт доступен для ИИ-краулеров, имеет понятный и описательный контент и хорошо структурирован, чтобы LLM могли легко понять, что вы предлагаете. Сайты, которые позиционируют себя сейчас для обнаружения ИИ, получат огромное преимущество по мере роста этого канала.

# Спасибо!

Спасибо, что приобрели этот курс и доверили нам свое время и деньги. Мы искренне надеемся, что он оказался для вас полезным и поможет вам создать что-то реальное.

Мы будем рады вашей обратной связи. **Присоединяйтесь к нашему сообществу в Discord** на apifreellm.com — это место, где мы общаемся, делимся новостями и помогаем друг другу.

Если у вас есть минутка, **напишите нам отзыв** и расскажите, что вы думаете. Что было наиболее полезным? Чего не хватило? Что можно улучшить? Мы искренне заинтересованы в вашей обратной связи — этот курс — живой проект, и мы хотим продолжать улучшать его на основе того, что **вам** действительно нужно.

**Увидимся в Discord. А теперь идите и создайте что-нибудь потрясающее.**