Первый день с Claude Code

Ты поставил агента. Что сделать сейчас, чтобы он работал в разы сильнее.

Кому: тем, кто только что впервые запустил Claude Code и хочет выжать из него максимум с первого дня.
Что внутри: настройка памяти агента, управление контекстом, как давать задачи, безопасность, расширения и принципы, которые мы вынесли из своей практики.
Как читать: по порядку в первый вечер, потом держать под рукой как чек-лист.

AirTap · Кемерово · Гайд новичка · 2026

Главная мысль на одну страницу

Claude Code — это не «чат, который пишет код». Это **агент**: он сам читает твои файлы, запускает команды, ставит пакеты, правит проект и проверяет результат. Поэтому разница между «слабым» и «сильным» агентом — не в модели (она у всех одна), а в том, **как ты его настроил и как с ним разговариваешь**.

Сильный агент с первого дня держится на четырёх вещах:

ПАМЯТЬ	Файл `CLAUDE.md`, где записано, кто ты, что за проект и как с тобой работать. Агент читает его каждую сессию.
КОНТЕКСТ	Понимание, что у агента ограниченное «окно внимания», и привычка его не засорять.
ЗАДАЧИ	Умение ставить задачу так, чтобы агент сначала думал, потом делал, и итерировал.
ГРАНИЦЫ	Правила безопасности и точки отката, чтобы «мощный» не стало «опасным».

> *Не пытайся освоить всё сразу. Сделай разделы 1–3 в первый вечер — этого уже хватит, чтобы агент стал заметно полезнее. Остальное добирай по мере надобности.*

1. Первые 15 минут после установки

1 Проверь, что всё на месте

В терминале:

```
claude doctor
```

Покажет версию и предупредит, если чего-то не хватает. Если команда `claude` вообще не находится — перезапусти терминал или добавь её в PATH (на Mac: `export PATH="$HOME/.local/bin:$PATH"`).

2 Заведи рабочую папку и запусти агента

Не запускай агента «в чистом поле» — дай ему папку проекта (даже если проект — это просто «мои заметки про бизнес»).

```
mkdir my-project
cd my-project
claude
```

При первом запуске выбери вход **«Claude account with subscription»** (по подписке), а не API-ключ.

3 Дай агенту изучить проект

Внутри `claude` набери:

```
/init
```

Агент просканирует папку и сам создаст черновик файла `CLAUDE.md` — это его будущая память. Если папка пустая — ничего страшного, мы заполним её в следующем разделе вручную.

> **Не давай агенту команды по памяти из интернета вслепую.** Если видишь совет «вставь эту команду» — сначала пойми, что она делает. Это первое правило безопасности, к нему вернёмся в разделе 5.

2. CLAUDE.md — мозг твоего агента

Это **самая важная настройка вообще**. `CLAUDE.md` — обычный текстовый файл, который агент автоматически читает в начале каждой сессии. Что в нём напишешь — тем он и руководствуется. Пустой агент гадает; агент с хорошим `CLAUDE.md` — знает твой контекст.

Два уровня памяти

ГЛОБАЛЬНЫЙ — Файл `~/.claude/CLAUDE.md` — правила для **всех** проектов: кто ты, на каком языке отвечать, твой стиль, общие запреты.

ПРОЕКТНЫЙ — Файл `CLAUDE.md` в папке проекта — про **этот** проект: что за бизнес, какие файлы важны, как собирать/проверять.

Что туда писать

Думай про `CLAUDE.md` как про инструктаж нового сотрудника в первый день. Полезные блоки:

- **Кто я и чем занимаюсь** — короткий контекст про тебя и бизнес.
- **Как со мной общаться** — язык, тон, «сначала коротко суть, потом детали», «не лей воду».
- **Что за проект** — цель, ключевые папки/файлы, как запустить и проверить.
- **Правила и запреты** — что нельзя трогать, что всегда спрашивать.
- **Готовые команды** — как у тебя собирается/деплоится/тестируется проект.

> *Самый быстрый способ заполнить файл — попросить самого агента: «Задай мне 7 вопросов про мой бизнес и проект и собери из ответов CLAUDE.md». Дальше правишь руками.*

Золотое правило

`CLAUDE.md` — живой документ. Каждый раз, когда агент сделал не то, потому что **чего-то не знал** — допиши это в файл. Через неделю таких правок агент будет понимать тебя с полуслова. Это и есть «прокачка»: ты растишь не модель, а её инструктаж.

3. Контекст: почему агент «тупеет» к концу сессии

У агента есть **окно контекста** — сколько текста он одновременно «держит в голове» (файлы, твои сообщения, результаты команд). Окно большое, но не бесконечное. Когда оно забивается — ответы становятся хуже, агент забывает, что было в начале.

Три привычки, которые это решают

1 Следи за заполнением

Команда `/context` показывает, сколько окна занято. Если под завязку — пора почистить (ниже). Многие ставят плагин статусной строки, который показывает заполнение и остаток лимита прямо внизу экрана.

2 Начинай новую задачу с чистого листа

Закончил одну тему, переходишь к другой — набери `/clear`. Это сбрасывает контекст. Не тащи мусор прошлой задачи в новую — это главная причина, почему агент «поглупел».

3 Сжимай длинные сессии

Если задача длинная и контекст распух, но бросать нельзя — `/compact`. Агент свернёт историю в краткую сводку и продолжит с ней. Можно подсказать, что важно сохранить: `/compact оставь решения и открытые задачи`.

> **Не держи одну бесконечную сессию на всё.** Одна сессия — одна задача. Это и быстрее, и дешевле по лимитам, и качество ответов выше.

> *Если что-то важное всплыло в разговоре и должно жить дольше сессии — не надейся на контекст, запиши это в `CLAUDE.md`. Контекст забывается, файл — нет.*

4. Как ставить задачи, чтобы получалось

Сначала план, потом работа

Для всего, что сложнее одной правки, проси агента **сначала составить план, а не бросаться делать**. В Claude Code для этого есть режим планирования (включается двойным `Shift+Tab` — агент исследует и предлагает план, ничего не меняя). Ты смотришь план, правишь, и только потом он работает. Это спасает от «сделал не то, что я хотел, но во всех файлах сразу».

Конкретика вместо общих слов

СЛАБО	СИЛЬНО
«сделай мне сайт»	«одностраничный сайт-визитка для моей кофейни: блок-герой, меню, карта, контакты; тёмная тема»
«почини, не работает»	«при отправке формы ничего не происходит, в консоли ошибка X — найди причину и почини»

Итерации важнее идеала

Не жди от первого ответа совершенства. Лучше быстрый черновик → ты смотришь → «теперь поправь вот это». Три коротких круга обратной связи дают результат лучше, чем одна гигантская инструкция на старте. Агент силён именно в диалоге.

Заставляй проверять себя

После работы проси: **«проверь, что получилось — собери проект / запусти / покажи результат»**. Агент умеет сам запускать то, что сделал, и ловить свои ошибки. Не принимай «готово» на слово — проси доказательство, что работает.

> *Привычка, которая экономит часы: перед большим изменением — «покажи план», после — «проверь результат». Думать до и проверять после.*

5. Границы: безопасность и откат

Мощный агент без правил — это риск. Две вещи, которые надо сделать в первый день.

Правило источников (вставь в CLAUDE.md)

Агент может сам ставить пакеты и запускать скрипты. Чтобы он не притащил вирус или скам, дай ему правила. Скопируй блок в `CLAUDE.md`:

```
## Безопасность: проверка источников
• Не скачивай и не запускай файлы, пакеты и скрипты из неизвестных источников.
• Перед установкой (npm/pip/brew/winget, curl…|bash, .dmg/.exe) проверь репутацию: офиц.
  сайт, живой GitHub, известность. Похоже на скам — остановись и спроси.
• Остерегайся тайпсквоттинга: имя пакета может отличаться на букву.
• Не вставляй мои пароли, ключи, seed-фразы во внешние сервисы.
• Трудно-откатываемое действие (установка, удаление, запуск с правами админа, отправка
  данных наружу) — сначала покажи и спроси.
• При сомнении «безопасно ли это» — остановись и спроси меня.
```

Точка отката всегда

Главный принцип нашей практики: **ломать можно — если есть откат**. Перед рискованным изменением — бэкап или коммит в git. Тогда любой эксперимент безопасен: не вышло — откатился.

```
git init # один раз в проекте
git add -A
git commit -m «рабочая точка» # перед каждым крупным шагом
```

Не разбираешься в git — попроси агента: «закоммить текущее состояние как точку отката перед изменением». Он сделает.

Разрешения

Чтобы агент не спрашивал подтверждение на каждую мелочь, но и не делал лишнего — настрой разрешения под себя командой `/permissions` (или через файл `settings.json`). Безопасные операции (чтение, поиск) разреши, потенциально опасные (удаление, отправка наружу) — оставь с подтверждением.

6. Чем усилить агента дальше

Это не на первый вечер, но знай, что есть — будешь подключать по мере роста.

1 Субагенты — делегирование

Агент может запускать **других агентов** под отдельные подзадачи параллельно (исследовать, проверить, собрать). Полезно, когда работа большая: «раздели на части и пройди параллельно». Тебе не надо ничего ставить — попроси словами.

2 MCP — подключение внешних инструментов

Протокол MCP даёт агенту доступ к внешним системам: база данных, GitHub, твоя CRM, поиск. Команда `/mcp` показывает подключённое. Так агент перестаёт быть «слепым» и начинает работать с твоими реальными данными.

3 Скиллы и слэш-команды

Свои повторяемые сценарии можно завернуть в команду — например `/отчёт` или `/пост`. Один раз описал — дальше вызываешь одной строкой. Экономит на рутине.

4 Хуки — автоматизация

Хуки запускают твои действия автоматически на события («после каждой правки — прогони проверку»). Это уже продвинутое, но именно хуки превращают агента в настоящий конвейер.

> *Не гонись за расширениями ради расширений. Сначала выжми базу (разделы 1–5). Подключай инструмент тогда, когда упёрся в конкретную нехватку, — а не «на всякий случай».*

7. Принципы, которые мы вынесли из практики

Это не про кнопки, а про подход. То, что реально отличает тех, у кого агент «летает».

ИТЕРАЦИИ	Лучше быстро выкатить черновик и улучшить по обратной связи, чем бесконечно полировать в голове. Движение важнее идеала.
РАЗВЕДКА	Перед тем как делать с нуля — спроси агента, кто уже решил это лучше и какими инструментами. Не изобретай велосипед.
САМОАУДИТ	Всегда проверяй результат перед тем, как принять. Код — собери и запусти. Текст — перечитай. «Готово» без проверки — не готово.
ОТКАТ	Рискуй смело, но всегда имей точку возврата. Бэкап перед экспериментом — и можно пробовать что угодно.
НЕ СТОЙ	Закончил задачу — бери следующую. Нет задач — улучшай то, что есть. Агент полезен ровно настолько, насколько ты его нагружаешь.

Агент — это усилитель. Он умножает того, кто им управляет. Поэтому вкладывайся не в «секретные промпты», а в ясность: чётко скажи, что хочешь, дай контекст, проверь итог.

8. Частые грабли первого дня

- **«command not found»** после установки — перезапусти терминал; не помогло — поправь PATH или перелогинься.
- **Опечатка в команде запуска** — это `claude` (к-л-а-у-д-е), а не `cloude`/`claude`/`cloud`.
- **Агент «поглупел» к вечеру** — забит контекст. `/clear` между задачами, `/compact` внутри длинной.
- **Сделал не то, что хотел** — давал общую формулировку. Конкретизируй и проси сначала план.
- **Вошёл не в тот аккаунт** — при логине в браузере был залогинен другой аккаунт Claude. Проверь, что открыт нужный, или используй приватное окно.
- **Страшно, что что-то сломает** — это не повод не пробовать. Сделай точку отката (git-коммит) и экспериментируй спокойно.

9. Чек-лист первого дня

- [ ] Запустил `claude doctor`, версия в порядке
- [ ] Завёл рабочую папку проекта, запускаю агента в ней
- [ ] Сделал `/init`, появился черновик `CLAUDE.md`
- [ ] Заполнил `CLAUDE.md`: кто я, что за проект, как со мной общаться
- [ ] Создал глобальный `~/.claude/CLAUDE.md` со своим стилем и языком
- [ ] Вставил в `CLAUDE.md` блок правил безопасности
- [ ] Сделал первый git-коммит как точку отката
- [ ] Знаю `/clear`, `/compact`, `/context` и когда их жать
- [ ] Попробовал режим плана (двойной `Shift+Tab`) на одной задаче
- [ ] Дал агенту первую реальную задачу и проверил результат

Прошёл чек-лист — у тебя уже не «голый» агент, а настроенный помощник.
Дальше — практика: чем больше реальных задач прогонишь через него в первую неделю, тем быстрее он станет незаменимым. Застрял — спроси в форуме, у нас все через это прошли.

AIRTAP · СООБЩЕСТВО · 2026