Стартовый набор Claude Code

Как работать с ИИ-агентом и не верить ему на слово: сторожа, субагенты, ворота и наши настоящие ошибки.

Этот набор собран из нашей работы над проектом AEVION, где несколько окон Claude Code ведут код, письма и замеры параллельно. Здесь нет теории про «идеальный промт». Только то, что мы применяем сами, и то, на чём сами ошиблись.

Главная мысль одна: **правило в тексте агент может забыть, а проверку в коде не забудет.** Всё остальное выводится из неё.

Что внутри

1. Файл правил: что туда писать
2. Хуки-сторожа и рабочий пример
3. Как проверить, что сторож работает
4. Субагенты: когда и как их звать
5. Ворота: когда агент обязан остановиться
6. Наши ошибки и как мы их теперь ловим
7. Список на первый день

Подготовлено с ИИ, факты сверены с нашими файлами. Это не официальный материал Anthropic. Результат зависит от вашей задачи, никакого дохода набор не обещает. Код здесь наш, проверяйте его у себя, прежде чем ставить.

1. Файл правил

Claude Code читает файл `CLAUDE.md` при старте. Это ваш способ сказать один раз то, что иначе пришлось бы повторять в каждом разговоре.

Что у нас окупилось

- **Цель и признак готовности.** Перед блоком работы агент пишет одну строку: какую цель двигает и по какому проверяемому признаку всё готово. Нет ответа — не начинает.
- **Метки уверенности.** Утверждение о внешнем мире идёт с меткой: точно (есть прибор), вероятно или догадка. Число — с прибором и датой.
- **Плохое первой строкой.** Иначе провал тонет в конце длинного отчёта.
- **Что только за человеком.** Пароли, платежи, подписи, цены. Этот список короткий и написан прямо.
- **Запрет молчаливых отказов.** Ошибка должна быть видна, а не превращаться в пустой ответ.

Чего туда не писать

- Длинных рассуждений: файл читается каждый раз и съедает контекст. Мы держим свой в пределах 12 КБ, а уроки выносим в отдельный архив.
- Секретов и ключей.
- Правил, которые нужно помнить в момент действия. Для них нужен сторож, об этом дальше.

Почему одного файла мало

Наш замер: кириллическое имя переменной в bash тихо ломает команду. За один день это случилось 11 раз в двух окнах, хотя правило было записано в файле дважды. Помнить надо в момент набора команды, а не в момент чтения файла. После этого появился сторож.

2. Хуки-сторожа

Хук — это ваша программа, которую Claude Code запускает на событие. Самое полезное событие — `PreToolUse`: хук видит команду до её выполнения и может её запретить.

Наши сторожа и причины

- **Перезагрузка.** Ноутбук дважды перезагрузился среди рабочего дня; виноват был Windows Update, но решение одно: машину гасит только человек, из какого бы источника ни пришла команда.
- **Выкатка.** За один день пять сессий выкатились в один сервис. Каждая прошла успешно и унесла работу предыдущей. Теперь хук спрашивает, чьё там сейчас.
- **Кириллица в переменных.** Предупреждает о случае выше.
- **Темп.** Считает выкатки и обращения к GitHub по всем окнам. За сутки у нас набралось не меньше ста выкаток сайта при суточном потолке площадки в сто: упрись в него, и не выкатится никто.

Рабочий пример сторожа

Файл `~/.claude/hooks/no-reboot.js`. Мы прогнали его на шести командах и битом вводе: запрещает три опасные, пропускает три обычные, на битом вводе молчит.

```
#!/usr/bin/env node
let raw = '';
process.stdin.on('data', (d) => (raw += d));
process.stdin.on('end', () => {
  let command = '';
  try {
    command = JSON.parse(raw).tool_input?.command || '';
  } catch {
    return; // непонятный ввод: не блокируем
  }
  if (/\b(shutdown|reboot|Restart-Computer|Stop-Computer)\b/i
      .test(command)) {
    process.stdout.write(JSON.stringify({
      hookSpecificOutput: {
        hookEventName: 'PreToolUse',
        permissionDecision: 'deny',
        permissionDecisionReason:
          'Перезагрузка только с согласия человека.',
      },
    }));
  }
});
```

Подключается в `~/.claude/settings.json` (путь подправьте под свою систему):

```
{ "hooks": { "PreToolUse": [ {
  "matcher": "Bash|PowerShell",
  "hooks": [ { "type": "command",
    "command": "node ~/.claude/hooks/no-reboot.js",
    "timeout": 10 } ]
} ] } }
```

Это простой пример: команду, спрятанную внутри скрипта, он не увидит. Защиту от всего он не даёт.

3. Проверьте сторожа

Сторож, которого не проверяли, хуже его отсутствия: вы думаете, что защищены.

> **Нужен положительный контроль.**
> Наш сторож темпа законно пропускает любой пуш, пока предел не исчерпан. Поэтому «пропустил» ничего не доказывает. Сначала покажите случай, который он обязан запретить, и убедитесь, что запрет сработал.

> **Смотрите на правильный сигнал.**
> Отказ приходит в JSON на стандартный вывод, а код выхода при этом 0. Проба, которая смотрит только на код, принимает каждый отказ за пропуск.

> **Ищите обход.**
> Сторож смотрит на текст команды, поэтому проверьте и непрямые пути: то же действие через скрипт или другую оболочку.

Минимальная проверка

```
echo '{"tool_input":{"command":"shutdown /r /t 0"}}' \
  | node no-reboot.js
# ждём: permissionDecision "deny"

echo '{"tool_input":{"command":"git status"}}' \
  | node no-reboot.js
# ждём: пустой вывод
```

Пишите файлы с регулярками аккуратно

У нас при записи через командную строку двойной обратный слэш схлопнулся в один, и `\b` в регулярке стал служебным символом. Она молча перестала находить часть слов. Теперь такой файл пишем инструментом записи и сразу прогоняем самотест.

4. Субагенты

Субагент — отдельный агент с чистым контекстом. Он знает только то, что вы написали ему в задании, и вы видите только его итоговый отчёт.

Когда зовём

- Механика: собрать данные, прочитать много страниц, прогнать проверку. Основное окно получает вывод, а не гору файлов.
- Свежий взгляд: деньги, вход, безопасность. Отдаём код, а не свой вывод, иначе он просто подтвердит вашу версию.

Когда не зовём

Если дела на пару команд и вы знаете, где искать. Субагент стоит дороже, чем кажется: обе передачи теряют детали.

Что в задании

- Цель и порог: что считается готовым.
- Какие файлы читать и что уже исключено.
- Запреты: что нельзя трогать, публиковать и менять.
- Правило про числа: нет открытого источника — так и напиши.

> **Цифру субагента проверяйте сами.**
> Наш субагент принёс счётчик заказов на бирже фриланса, и на нём держался целый рейтинг способов заработка. Счётчик оказался общим для продавца, а не для конкретного товара, и рейтинг пришлось пересобирать. В другой раз цитаты были верны, но пришли через пересказ: мы открыли все страницы сами и только потом передали дальше.

Правило: если на числе держится решение, его перепроверяет тот, кто решает.

5. Ворота

Ворота — проверка перед необратимым действием: отправкой, выкаткой, публикацией. Они срабатывают кодом, а не надеждой.

- **Код выхода не равен 0 — стоп.** Без исключений и без «наверное, ничего страшного».
- **Код выхода смотрим без конвейера.** В конструкции с `|` обычно виден код последней команды, а не той, что упала.
- **«Работает» доказывает живой замер.** Ответ 200 от адреса проверки здоровья ещё не значит, что работает приложение.
- **«Идёт» пишем только с числом из лога.** Наша цепочка отправки однажды умерла от сетевой ошибки, и письма простояли до утра, хотя в отчёте значились «идущими».
- **«Не отправлено» — только после сверки с папкой отправленных.**
- **Ровный ноль сначала проверяет прибор.** Наш `grep -i` оказался слеп к кириллице и уверенно показывал «ничего нет».
- **Согласие не переносится.** «Да» на одну пачку писем не значит «да» на другую.

Настройки разрешений

Не обходите защиту Claude Code и не просите агента менять собственные разрешения. Если что-то нужно часто, человек разрешает это сам и осознанно. Если проверка отказала, не передавайте ту же работу другому окну: это тот же обход.

6. Наши ошибки

«Люди с YouTube» оказались обходчиком.
Прибор показывал живых людей с роликов. Мы сверили время: все 6 сессий с метками роликов за две недели пришли через 4–45 секунд после выхода своего ролика. Отклик на странице обходчика не выдал. Теперь сверяем время захода со временем публикации.
Ловим: новым правилом в счётчике и контрольными днями с известным ответом.

«Referrer не передаётся» — было ложью.
Агент смотрел только во вложенное поле и уверенно сказал, что данных нет. Они лежали на уровень выше.
Ловим: перед «данных нет» печатать все ключи.

Правка фразы пропустила 10 писем.
Искали фразу в одну строку, а в тексте она была с переносом.
Ловим: проверка по уже готовому результату, а не только по исходнику.

Потерянный слэш в проверке адреса.
В регулярке `\s` стал буквой `s`: адрес с буквой s не проходил, а с пробелом проходил. Тесты при этом были зелёные. Поймал другой агент при приёмке.
Ловим: тест на неудобные значения и приёмку свежими глазами.

Агент принял не ту роль.
После очистки контекста агент решил по косвенным следам, что он другое окно, и начал проверять чужие задачи. Сама роль записана в строке запуска.
Ловим: определять, кто ты, по прибору, а не по догадке.

7. Список на первый день

1. Создайте короткий `CLAUDE.md`: цель, признак готовности, метки уверенности, что только за человеком.
2. Выберите одно действие, которое вас пугает, и поставьте на него сторожа.
3. Проверьте сторожа положительным контролем и попробуйте его обойти.
4. Дайте агенту задачу с проверяемым признаком готовности и попросите показать доказательство, а не слова.
5. Одну механическую задачу отдайте субагенту, а главную цифру из его отчёта проверьте сами.
6. Первую ошибку агента запишите одной строкой: что случилось и какая проверка теперь это ловит.

Если поймаете у себя ошибку, которой здесь нет, напишите мне: добавлю в следующую версию.

Про наш проект

Этими приёмами мы строим AEVION. Там же есть DevHub: сайт по описанию, от слов до живого адреса, без кода.

aevion.app/devhub?c=ig-kit-1111

Подготовлено с ИИ. Не официальный материал Anthropic. Claude и Claude Code — продукты Anthropic.