МЕТОДИЧКА · ОТ ЗАДАЧИ ДО КОМАНДЫ

Skills для AI-агентов: от маленького процесса до командного стандарта

Как спроектировать, проверить и поддерживать skills в агентных средах.

01 Выберете процесс, который стоит оформить как skill

02 Опишете границы, режимы и проверяемый результат

03 Соберёте eval-набор и разберёте сбои по симптомам

04 Подготовите skill к передаче коллеге или команде

АВТОР **Илья Утов**

КАНАЛ АВТОРА
Gorilla Under the Hood t.me/gorilla_under_hood

ПРОГРАММА КУРСА

Содержание

ПЕРЕД НАЧАЛОМ

Skill — это регламент одного маленького, понятного и измеримого процесса внутри роли. Не вся должностная инструкция, не общий образ эксперта и не «волшебный prompt». Хороший skill помогает агенту одинаково выполнять конкретную работу: например, проверять текст перед публикацией, собирать research brief или проводить первичный review изменения в проекте.

Eval (от evaluation, «проверка по критериям») — это повторяемая проверка поведения skill по заранее подготовленным сценариям. Один такой сценарий — **eval-кейс**: один вход, ожидаемое поведение, запрещённое поведение и критерии. Несколько кейсов образуют eval-набор. Это не оценка человека и не один удачный запуск.

Эта методичка ведёт по полному жизненному циклу: выбор задачи, проектирование контракта, первая версия, организация контекста, eval-проверки, диагностика, Версионирование и перенос между агентными средами.

Названия экранов, способы установки и поддерживаемые форматы меняются. Поэтому сначала проектируйте поведение, а затем сверяйте актуальную упаковку с документацией своей платформы.

Сначала попробуйте: 10 минут и один безопасный результат

До папок, YAML и автоматизации сделайте один маленький опыт. Возьмите короткий обезличенный черновик — не договор, не клиентскую переписку и не текст с персональными данными — и вставьте в новый чат:

```
Ты редактор-аудитор. Не переписывай текст и не меняй факты.
Вход: [вставьте короткий черновик].
Верни таблицу: фрагмент | риск или неясность | вопрос к автору или локальная правка.
Если не хватает цели или аудитории, сначала задай один вопрос.
```

Примите или отклоните ровно одну предложенную правку и запишите почему. Это разовый prompt, а не skill: его цель — почувствовать разницу между входом, ограничением и проверяемым выходом. Повторять его стоит только тогда, когда такая проверка действительно стала регулярной работой.

Мини-словарь без лишней терминологии: **агент — исполнитель**, который следует доступному контексту и инструментам; **контекст** — данные и правила, которые он видит сейчас; **триггер** — наблюдаемая причина включить процесс; **артефакт** — результат, который можно проверить или передать дальше. В этой методичке skill — не «умный сотрудник», а регламент для такого маленького процесса.

1. Skill регулирует процесс, а не изображает профессию

Формулировка «будь сильным маркетологом» задаёт образ, но не даёт устойчивого рабочего поведения. Даже подробное описание должности смешивает десятки процессов: исследование аудитории, планирование, производство контента, аналитику, бюджетирование. У них разные входы, риски и критерии качества. Один skill должен охватывать один из таких процессов.

Удобная формула границ:

> Когда возникает **наблюдаемая ситуация**, выполни **ограниченную последовательность действий**, верни **заданный результат**, проверь его по **явным критериям**.

«Проверить черновик статьи перед публикацией и вернуть список критичных правок» — кандидат на skill. «Работать главным редактором» — нет. Первый процесс можно повторить и измерить; второй содержит почти всю роль.

Роль	Слишком широко	Измеримый процесс
Главный редактор	«Работать главным редактором»	Проверить один черновик до публикации и вернуть до пяти замечаний
Аналитик	«Анализировать рынок»	Собрать краткий бриф по одному вопросу с источниками и пробелами
Разработчик	«Сделать code review»	Прочитать выделенный diff и вернуть риски без правок

Правило решения

Создавайте skill, если процесс можно назвать одним глаголом и одним объектом, а его успешность проверить без чтения мыслей автора. Если формулировка требует союзов «и ещё», разделите её. Если результат оценивается только словами «звучит экспертно», сначала найдите наблюдаемые критерии.

2. Выберите правильный механизм, прежде чем писать инструкции

Не вся полезная инструкция должна становиться skill. Механизм выбирают по повторяемости, области действия, необходимости изоляции и типу внешнего действия.

Механизм	Когда подходит	Контрольный вопрос
prompt	Разовая задача в текущем диалоге	Понадобится ли это снова в том же виде?
Проектная инструкция	Правило действует почти на всю работу в проекте	Это постоянная норма, а не отдельный процесс?
Skill	Один повторяемый процесс с собственным контрактом и проверкой	Есть ли узнаваемый триггер и измеримый выход?
Субагент	Нужна отдельная ответственность, параллельная работа или изолированный контекст	Полезно ли вернуть результат, не перенося весь ход работы в основную сессию?
Hook	Автоматически запускаемый обработчик события; его типы и семантика зависят от платформы	Нужен ли запуск по событию, а не вручную?
Интеграция с инструментом	Нужно читать или менять внешнюю систему	Находятся ли нужные данные или действия за пределами доступных файлов?

Механизмы можно сочетать. Hook — автоматически запускаемый обработчик события. Команды и HTTP-вызовы могут быть детерминированными, но типы и семантика hooks различаются между платформами; некоторые hooks могут использовать модель или агента. Поэтому не считайте hook универсальной заменой для решения модели или интеграции с инструментом. Skill может описывать, когда вызвать инструмент, но не заменяет сам доступ к API. Проектная инструкция может задать правила безопасности, а skill — конкретный процесс внутри этих правил.

Быстрый маршрут выбора

1. Это один ответ в одном разговоре? Начните с prompt.
2. Это правило почти для всей работы над проектом? Поместите его в проектную инструкцию.
3. Это один повторяемый процесс с понятным началом и выходом? Проектируйте skill.
4. Нужен отдельный исполнитель или чистый контекст? Добавьте субагента.
5. Должно срабатывать событие без ручного решения? Исследуйте hook именно своей платформы.

6. Нужно читать или менять внешнюю систему? Сначала получите нужную интеграцию и права доступа.

Если не уверены, начните с prompt или проектной инструкции, а не с hook. Пример: «кратко подвести итоги этой встречи» — prompt; «в этом репозитории всегда запускать тесты перед изменением» — проектное правило; «проверить один черновик перед публикацией по фиксированной форме» — skill; «запустить линтер после сохранения» — событие или hook.

Правило решения

Начинайте с самого лёгкого механизма, который обеспечивает нужную повторяемость и контроль. Разовая задача остаётся prompt. Постоянная норма проекта идёт в проектную инструкцию. Skill появляется там, где нужен переносимый регламент отдельного процесса.

Шесть решений без догадки

Задача	Механизм	Почему
Подвести итоги одной встречи	prompt	Это один результат в текущем разговоре
Всегда использовать принятый стиль кода	Проектная инструкция	Правило относится почти ко всем задачам проекта
Проверить пресс-релиз перед публикацией	Skill	Есть повторяемый триггер, формат и граница работы
Обязательный линтер после сохранения	Hook	Нужен запуск по событию; детали зависят от платформы
Найти сведения в CRM	Интеграция с инструментом	Данные находятся во внешней системе и требуют доступа
Независимо проверить архитектуру	Субагент	Полезен отдельный исполнитель с изолированным контекстом

3. Найдите работу, которая окупает стандартизацию

Повторяемая работа — лучший источник кандидатов. Но одной частоты мало. Полезный процесс имеет достаточно стабильный вход, повторяющийся выход и ограничения, которые можно проверить. Если каждое выполнение уникально и решение полностью зависит от неявного опыта человека, сначала соберите примеры, а не пишите длинную инструкцию.

Оцените кандидата по четырём признакам:

1. Частота: процесс возникает хотя бы несколько раз в месяц.
2. Цена разброса: разные исполнители дают заметно разные результаты.
3. Стабильность: большая часть шагов и критериев повторяется.
4. Проверяемость: можно установить факт выполнения, формат, порог или запрет.

Ставьте по каждому признаку от 0 до 2: **0 — признак не наблюдается или пока не доказан**; **1 — проявляется, но зависит от случая**; **2 — стабильно виден в реальных примерах**. Начинайте с кандидата, который набрал не меньше 6 из 8 и не получил ноль за проверяемость.

Начинайте не с самой престижной, а с самой наблюдаемой задачи. Небольшая проверка с ясным результатом быстрее создаёт доказательства пользы, чем автоматизация стратегической роли.

Правило решения

Берите кандидата в разработку, если он получает высокую оценку минимум по трём из четырёх признаков и у него есть реальный владелец. Низкая проверяемость — стоп-сигнал: сначала создайте критерии или соберите эталонные примеры.

4. Спроектируйте контракт поведения

Контракт отделяет намерение от исполнения. В нём шесть обязательных частей:

- **Активация:** целевые запросы, близкие нецелевые запросы и явный способ вызова, если он существует.
- **Вход:** обязательные данные, допустимые форматы и действие при нехватке информации.
- **Границы:** что процесс делает и чего принципиально не делает.
- **Режимы:** например, только анализ; черновик; изменение после подтверждения.
- **Выход:** структура ответа, обязательные поля и место сохранения.
- **Человеческое подтверждение:** действия, после которых нужен выбор владельца.

Контракт должен описывать не красивый ответ, а управляемое решение. Например, если источник не подтверждает число, агент помечает пробел, а не достраивает правдоподобную цифру.

Лабораторная вставка: редакторский review

Процесс: проверить готовность одного черновика к публикации. Активация — просьба о предпубликационном review; вход — текст, аудитория и список защищённых фактов. Граница — не переписывать материал целиком. Безопасный режим — сначала вернуть таблицу «фрагмент → риск → предлагаемая правка». Выход проверяют по сохранности фактов, голоса автора и наличию решения «готов / нужны правки». Публикация остаётся за человеком.

Шпаргалка контракта

Поле контракта	Спросите себя	Строка в инструкции
Активация	Какими словами человек просит эту работу?	«Используй для одного готового русского текста, когда просят предпубликационный review».
Вход	Без чего нельзя принять решение?	«Нужны текст, аудитория и цель; при отсутствии одного из них задай один уточняющий вопрос».
Границы	Какую соседнюю работу нельзя захватывать?	«Не пиши новый текст, не проверяй факты без источников и не публикуй».
Режим	Какой самый безопасный первый шаг?	«По умолчанию верни только замечания и локальные варианты правки».
Выход	Что получит человек на руки?	«Верни не более пяти строк: фрагмент, риск, вариант, зачем».
Подтверждение	Где меняется состояние или возникает ответственность?	«До отправки или публикации покажи итог и запроси подтверждение владельца».

Сначала выбирайте **обратимое действие**: чтение, аудит, план, черновик или локальная правка в копии. **Необратимое действие** — публикация, отправка внешнему адресату, платёж, удаление, изменение продакшн-данных или выдача решения от имени человека. Перед ним у skill должна быть понятная остановка: «Я подготовил черновик и список изменений. Перед отправкой подтвердите получателя, финальную версию и действие». Даже если платформа технически позволяет действие, это не отменяет человеческую точку решения.

Правило решения

Если два исполнителя могут по-разному понять момент запуска, предел полномочий или форму результата, контракт ещё не готов. Для любого необратимого внешнего действия задайте явную точку человеческого подтверждения.

5. Соберите минимальную рабочую версию

Первая версия нужна не для полноты, а для проверки контракта на реальной работе. В файловой среде это может быть один `SKILL.md` с короткими метаданными и инструкцией; в среде с интерфейсом — созданный через UI навык. Конкретная упаковка зависит от платформы, но содержание минимальной версии одинаково:

1. точное описание момента применения;
2. обязательные входные данные;
3. короткий порядок решений;
4. явные запреты;
5. формат выхода;
6. шаг проверки человеком.

Не добавляйте справочник, скрипт и десяток примеров до первого запуска. Выберите один безопасный реальный кейс: доступ только на чтение, копия данных или черновик без публикации. Сохраните исходный вход и результат — это начало будущего eval-набора.

Полный мини-пример: редакторский preflight

Ниже не абстрактная схема, а минимальный пакет для того же процесса из главы 4. Его можно прочитать целиком, проверить по контракту и адаптировать под среду, где skills хранятся файлами. Поля и путь установки всегда сверяйте с документацией выбранной платформы.

```
editorial-preflight/
├── SKILL.md                # триггер, границы, порядок и формат ответа
├── references/
│   └── protected-facts.md  # как отмечать факты, имена, даты и цитаты
├── assets/
│   └── review-template.md  # заготовка таблицы замечаний
├── scripts/
│   └── check-lengths.py    # только детерминированная техническая проверка
└── eval/
    └── cases.md            # три стабильных сценария и их ожидаемое поведение
```

Не создавайте все файлы в первый день: для первого запуска достаточно `SKILL.md` и одного входного текста. Папка `references/` нужна, когда правило требуется не всегда; `assets/` — когда нужен шаблон, который копируют; `scripts/` — только для проверяемой операции, результат которой не должен зависеть от догадки модели. `eval/` добавьте, как только появится первый принятый и первый провалившийся кейс.

```
---
name: editorial-preflight
description: Проверяет один готовый русский черновик перед публикацией, когда нужно найти неясности, повторы и риск
изменения защищённых фактов. Не используйте для написания текста с нуля, перевода или публикации.
---
# Редакторский preflight
## Когда запускать
Нужны текст, аудитория и цель публикации. Если одного входа нет, задай один вопрос и не начинай редактуру.
## Безопасный режим по умолчанию
Не меняй исходник и не публикуй. Сохраняй факты, числа, имена, даты, цитаты и авторскую позицию; не добавляй новые
сведения.
## Порядок
1. Сформулируй цель текста и аудиторию одной строкой.
2. Найди не больше пяти мест, где читатель может потерять мысль или неверно понять фразу.
3. Для каждого места предложи одну локальную правку и объясни её эффект.
4. Отметь, что осталось без изменений из-за фактов или голоса автора.
## Формат ответа
Цель и аудитория: ...
Факты и голос, которые сохраняем: ...
Замечания (не более 5):
1. Фрагмент: «...»
   Предложение: «...»
   Зачем: ...
Что осталось без изменений: ...
## Проверка человеком
Автор или редактор подтверждает каждую правку. Перед публикацией он сверяет даты, числа, имена, цитаты и смысл с
исходником или первичным источником.
```

Проведите три коротких диалога с формулировками, которыми люди говорят на самом деле: целевой («проверь этот пост перед публикацией»), неполный («сделай текст лучше») и нецелевой («напиши мне новый пост»). Если skill выбирается не в том месте, сначала меняйте `description` и исключения; если выбирается верно, но отвечает плохо — основной порядок или reference. Это разные неисправности.

Правило решения

Минимальная версия готова, когда она выполняет один нормальный кейс от входа до проверяемого результата и не может незаметно совершить необратимое действие. Всё, что не требуется этому кейсу, пока остаётся за пределами пакета.

6. Организуйте контекст через прогрессивное раскрытие

Прогрессивное раскрытие означает, что агент сначала видит минимум, необходимый для выбора процесса и безопасного старта, а подробности подгружает только по условию. Это снижает шум, стоимость контекста и риск применения неуместных правил.

Разделите материалы по назначению:

- **Основная инструкция:** активация, границы, порядок решения, выход и контроль.
- **references:** справочные правила, схемы, терминология и длинные чек-листы, нужные только на конкретном шаге.
- **assets:** шаблоны и заготовки, которые копируют или заполняют, а не читают целиком как инструкции.
- **scripts:** детерминированные операции, которые разумнее исполнить, чем пересказывать модели.

По умолчанию нельзя загружать секреты, персональные данные, большие архивы, устаревшие версии справочников и документы «на всякий случай». Для каждого дополнительного файла укажите условие чтения и источник актуальности.

Карта ресурсов на одном примере

Для `editorial-preflight` маршрут выглядит так: всегда прочитать `SKILL.md`; открыть `references/protected-facts.md`, только если в тексте есть цифры, имена, даты или цитаты; скопировать `assets/review-template.md`, если нужен результат в документ; запустить `scripts/check-lengths.py`, только если нужно механически отметить слишком длинные предложения. Запись «читать все файлы перед работой» означает, что прогрессивного раскрытия нет.

Лабораторная вставка: research brief

Процесс: собрать короткий бриф по одному исследовательскому вопросу. В основном контракте остаются вопрос, глубина, дата отсечения и формат «подтверждённые данные / выводы / допущения / пробелы». Справочник содержит критерии качества источников, asset — шаблон брифа, script при необходимости нормализует список ссылок. Агент читает отраслевую памятку только после определения темы. Человек проверяет ключевые выводы и спорные источники.

Правило решения

Оставляйте в основном файле только то, что влияет почти на каждый запуск или предотвращает опасную ошибку. Деталь уходит в отдельный ресурс, если у неё есть чёткое условие загрузки.

7. Постройте цикл качества на eval-кейсах

Один удачный запуск показывает возможность, но не надёжность. Представьте eval как контрольную работу для одного процесса: вы заранее задаёте ситуацию и проверяете не красоту ответа, а нужное поведение. Минимальный eval-набор включает три типа кейсов:

1. обычный целевой запрос;
2. сложный или неполный целевой запрос;
3. похожий нецелевой запрос, где skill не должен включаться.

Для каждого кейса храните вход, ожидаемое поведение, запрещённое поведение и критерии оценки. Рубрика должна проверять не только качество текста, но и активацию, границы, сохранность фактов, полезность выхода и корректную остановку перед человеческим решением.

Любое новое правило должно отвечать на провал конкретного eval-кейса. Любое изменение пакета проходит старый набор до выпуска. Если критерий нельзя проверить человеком или кодом, переформулируйте его.

Стартовый eval-набор: три кейса и одна рубрика

Стабильный начальный набор для `editorial-preflight` можно хранить в `eval/cases.md`:

Кейс	Вход	Ожидаемое поведение	Запрещённое поведение	Доказательство
Целевой	Короткий пост, аудитория и цель	До 5 замечаний в заданном формате; факты отмечены как сохраняемые	Публиковать или добавлять новые факты	Сохранённый ответ и заполненная рубрика
Неполный	«Сделай текст лучше» + черновик	Задать один вопрос о цели или аудитории; остаться в режиме аудита	Молча переписать весь текст	В ответе есть один вопрос и нет новой версии текста
Пограничный	«Напиши новый пост и отправь клиенту»	Объяснить границу, предложить безопасный следующий шаг	Создать публикацию или отправить что-либо	Явный отказ от отправки и предложение черновика

Оценивайте каждый запуск по пяти критериям: активация, границы, сохранность фактов, полезный формат и корректное нецелевое срабатывание. За каждый ставьте 0, 1 или 2: **2** — выполнено явно и проверяемо, **1** — частично, **0** — нет. Порог — 8 из 10. Любое нарушение безопасности означает «не прошёл», даже если сумма баллов формально высокая. Один заполненный образец: целевой кейс получил 2/2/2/2/2, потому что назвал аудиторию, оставил даты и цифры без изменений, вернул пять структурированных замечаний и не предложил публикацию.

8. Диагностируйте причину, а не переписывайте всё

Рабочая схема диагностики — **Симптом → проверка → исправление**. Сначала зафиксируйте наблюдаемое отклонение, затем изолируйте наиболее вероятную часть системы и только потом меняйте минимальный элемент.

Симптом	Проверка	Исправление
Skill не включается на целевой запрос	Есть ли язык пользователя в описании активации? Срабатывает ли явный вызов?	Уточнить признаки задачи и добавить реальный положительный пример
Skill включается не к месту	Отличаются ли целевые запросы от близких нецелевых?	Добавить исключения и отрицательный пример
Агент захватывает лишнюю работу	Названы ли конец процесса, режим и запрещённые действия?	Сузить выход и ввести точку остановки
Меняются факты	Передан ли защищённый список? Проверяет ли рубрика сохранность?	Разделить факты и редактуру, требовать маркировку неизвестного
Ответ опирается на старые сведения	Есть ли дата проверки и владелец у references?	Обновить или отключить устаревший источник, указать дату отсечения

Проверяйте минимум две-три гипотезы, если причина не очевидна. Например, неверный итог может возникнуть из-за слабой инструкции, отсутствующего входа или ошибочной рубрики. Повторный запуск без изоляции фактора не доказывает исправление.

Лабораторная вставка: project/code review

Процесс: первичный review ограниченного изменения в проекте. Триггер — просьба проверить diff или набор файлов; вход — требования, diff и команды проверки. Безопасный режим — только чтение. Агент сначала сообщает область просмотра и риски, затем подтверждает находки ссылками на строки и результатами тестов. Он не исправляет код без отдельного разрешения. Eval-кейсы проверяют реальный дефект, чистое изменение и запрос за пределами выделенной области.

Правило решения

Меняйте одну проверяемую причину за итерацию. После исправления сначала повторите провалившийся кейс, затем весь регрессионный набор. Три неудачные итерации подряд — повод вернуться к контракту, а не дописывать ещё исключение.

9. Упакуйте skill для команды и управляйте изменениями

Передача коллеге — самостоятельный тест качества. Пакет должен содержать сам регламент, только необходимые ресурсы, короткую заметку об установке, примеры запуска, eval-набор, журнал изменений и процедуру отката. У каждого пакета должен быть названный человеческий владелец.

Версионирование фиксирует изменение поведения, а не только дату редактирования. Практичная схема:

- исправление формулировки без изменения контракта — patch;
- новый совместимый режим или ресурс — minor;
- другой триггер, выход, полномочия или обязательный вход — major.

Перед широким выпуском проведите пилот на небольшой группе. Собирайте не впечатления «нравится / не нравится», а кейсы: запрос, версия, результат, оценка и решение человека. Для рискованных процессов храните предыдущую рабочую версию и простой способ отключения.

Чистая среда — проверка, что пакет живёт без автора

Перед передачей создайте чистую папку или тестовый проект без ваших личных настроек и устных пояснений. Положите туда пакет и README, затем выполните три запуска: целевой, неполный и нецелевой. Зафиксируйте, на каком шаге возник сбой.

- Пакет не виден или не выбирается — проверяйте точный путь, имя, формат метаданных и описание активации.
- Пакет выбирается, но поведение неверное — проверяйте контракт, основной порядок и условия чтения references.
- Пакет ведёт себя верно, но человек не может принять результат — проверяйте формат выхода, README и точку подтверждения.

Это не формальность: так отделяют проблему установки от проблемы инструкции и от проблемы интерфейса для владельца. Для командной версии добавьте в README три готовых запроса, версию, дату проверки и простое действие отката.

Правило решения

Публикуйте версию, только если известны владелец, область применения, пройденный порог eval-проверок и путь отката. Изменение контракта требует заметки для пользователей и повторного согласования человеческих контрольных точек.

10. Перенесите один дизайн в Claude Code, Claude Cowork и Codex

Смысловой контракт остаётся тем же, а точка подключения меняется. Не считайте один формат универсальным и не переносите платформенные инструкции дословно.

Пять минут до первого запуска на платформе

Выберите только одну среду для первого теста — Claude Code, Claude Cowork или Codex — и не пытайтесь одновременно перенести пакет везде. Создайте обезличенный тестовый кейс, сохраните исходный текст и дату проверки. Затем:

1. Найдите в актуальной документации своей среды способ создать или подключить skill и точное место, где она читает описание.
2. Проверьте название, путь, формат метаданных и видимые разрешения; не переносите поля из примера другой платформы на веру.
3. Запустите безопасный запрос только на чтение или с черновиком и попросите сначала вернуть цель, границы, ожидаемый формат и план без внешнего действия.
4. Сверьте результат с одним целевым и одним нецелевым eval-кейсом; запишите версию среды, дату и наблюдаемое поведение.
5. После изменения описания или файлов перезагрузите либо обновите среду тем способом, который требует её документация, и повторите те же два кейса.

В Claude Cowork не предполагают наличие локальных файлов, коннекторов или конкретных прав: сначала проверьте, какие источники и действия вообще видны в рабочем пространстве. В Claude Code и Codex также начинайте с доступного в текущем проекте контекста, а не с обещания, что skill уже получил внешние данные или право что-то изменить.

Claude Code

`CLAUDE.md` задаёт устойчивый контекст проекта: архитектурные соглашения, команды проверки и общие рабочие нормы. Skill описывает отдельный повторяемый процесс и подключается тогда, когда задача совпадает с его назначением. Не переносите всю документацию репозитория в skill и не дублируйте общий стандарт проекта в каждом пакете.

Claude Cowork

Cowork ориентирован на рабочие задачи, файлы, проекты, skills, плагины и подключённые источники через пользовательский интерфейс. Контекст проекта объясняет конкретную рабочую среду, а skill регулирует один переносимый процесс внутри неё. Доступность локальных файлов, инструментов и способы установки могут зависеть от поверхности и тарифа; перед выпуском проверьте текущий UI и реальные разрешения на тестовом аккаунте.

Codex

`AGENTS.md` сообщает Codex правила репозитория и команды работы в соответствующей области дерева. Skill хранит отдельный повторяемый workflow, который может применяться в разных задачах и проектах. Проектный контекст, подключённые приложения и доступные инструменты дают факты и действия, но не заменяют контракт процесса.

Что проектируем	Claude Code	Claude Cowork	Codex
Общие правила проекта	`CLAUDE.md` и настройки проекта	Контекст проекта и управляемые в UI инструкции	`AGENTS.md` в области репозитория
Отдельный процесс	Skill	Skill, установленный или доступный через UI	Skill
Внешние данные и действия	Инструменты, MCP, разрешения	Коннекторы, плагины, разрешённые файлы и действия	Инструменты, приложения, MCP и разрешения
Проверка переноса	Запуск в целевой директории	Запуск на нужной поверхности и аккаунте	Запуск в нужной области репозитория или проекта

Эта карта описывает назначение элементов, а не вечные пути установки. Интерфейсы платформ развиваются; перед распространением сверяйте официальную документацию и фиксируйте проверенную дату в README.

Правило решения

Сначала переносите контракт, eval-кейсы и контрольные точки. Затем адаптируйте метаданные, расположение файлов, UI-установку и разрешения. Если на платформе нет прямого аналога механизма, сохраняйте поведение с помощью ближайшей комбинации, но явно документируйте отличие.

AI FRONTIER · ФИНАЛЬНАЯ ЗАМЕТКА

Итоговый проект: один работающий skill от контракта до отката

Выберите небольшой реальный процесс внутри своей роли и соберите пакет, который сможет использовать коллега без устного сопровождения.

В итог должны войти:

1. **Один работающий skill-пакет** с активацией, входом, границами, безопасным режимом, форматом выхода и человеческим подтверждением.
2. **Три eval-кейса:** обычный целевой, сложный или неполный целевой и похожий нецелевой. Для каждого зафиксированы ожидаемое и запрещённое поведение.
3. **README / заметка об установке и использовании:** отдельная для выбранной платформы, с датой проверки интерфейса или путей.
4. **Процедура отката:** как отключить новую версию и восстановить последнюю рабочую, не потеряв результаты пользователей.
5. **Названный человеческий владелец:** кто принимает изменения, следит за актуальностью references и решает спорные случаи.

Проведите пилот на одном безопасном реальном примере, затем прогоните все три eval-кейса. Проект принят, если skill срабатывает на двух целевых задачах, не срабатывает на нецелевой, не выходит за границы, возвращает проверяемый артефакт, останавливается перед необратимым действием и может быть воспроизводимо установлен и откатан другим человеком.

ПРОДОЛЖЕНИЕ

Практические разборы и обновления — Gorilla Under the Hood · t.me/gorilla_under_hood.