Полное руководство по созданию навыков для Claude

Claude

Переведено t.me/aivkube

Содержание

Введение

Навык - это набор инструкций **(https://claude.com/blog/skills)**, упакованных в простую папку, который обучает Claude выполнять конкретные задачи или рабочие процессы. Навыки — один из самых мощных способов персонализации Claude под ваши конкретные нужды. Вместо того чтобы каждый раз заново объяснять свои предпочтения, процессы и экспертные знания в определённой области, навыки позволяют обучить Claude один раз и получать выгоду каждый раз.

Навыки особенно эффективны при повторяющихся рабочих процессах: создании frontend-дизайнов по техническим заданиям, проведении исследований с последовательной методологией, подготовке документов в соответствии со стилевыми рекомендациями вашей команды или организации многоэтапных процессов. Они отлично сочетаются со встроенными возможностями Claude, такими как выполнение кода и создание документов. Для разработчиков интеграций MCP навыки добавляют ещё один мощный уровень, помогая превратить прямой доступ к инструментам в надёжные и оптимизированные рабочие процессы.

Данное руководство охватывает всё, что необходимо знать для создания эффективных навыков — от планирования и структуры до тестирования и распростране-ния. Независимо от того, создаёте ли вы навык для себя, своей команды или сообщества, здесь представлены практические шаблоны и примеры из реальной практики.

Что вы узнаете:

- Технические требования и лучшие практики построения структуры навыков
- Шаблоны для автономных навыков и рабочих процессов с использованием MCP
- Шаблоны, которые доказали свою эффективность в различных случаях применения
- Как тестировать, улучшать и распространять ваши навыки

Для кого это:

- Разработчиков, которые хотят, чтобы Claude последовательно выполнял определённые рабочие процессы
- Продвинутых пользователей, которые хотят, чтобы Claude выполнял определённые рабочие процессы
- Команд, стремящихся стандартизировать работу Claude по всей своей организации

Два пути в этом руководстве

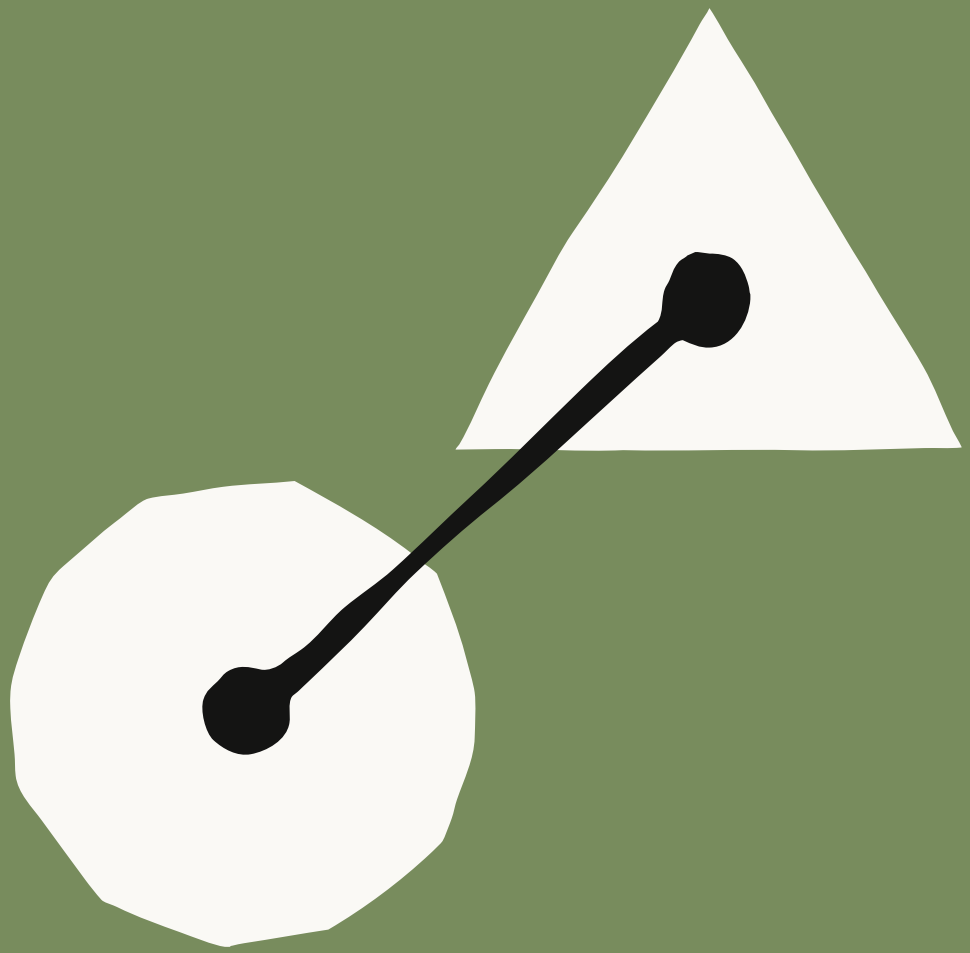
Создаёте автономные навыки?

Сосредоточьтесь на основах, планировании и дизайне, а также категориях 1–2.

Улучшаете интеграцию MCP?

Раздел «Навыки + MCP» и категория 3 предназначены для вас.

Оба пути имеют одинаковые технические требования, но вы выбираете то, что актуально для вашего случая. В конце вы сможете создать функциональный навык за один раз. Ожидайте, что создание и тестирование первого рабочего навыка с помощью конструктора навыков займёт около 15–30 минут.



Глава **1**

Основы

Глава **1**

Основы

Что такое навык?

Навык — это папка, содержащая:

- **SKILL.md (обязательно):** Инструкции в формате Markdown с YAML-метаданными
- **scripts/ (**необязательно): Исполняемый код (Python, Bash и др.)
- **references/** (необязательно): Документация загружается по мере необходимости
- **assets/** (необязательно): Шаблоны, шрифты, иконки, используемые в выводе

Основные принципы проектирования

1. Постепенное раскрытие

Навыки используют трёхуровневую систему:

a. **Первый уровень (YAML-метаданные):** всегда загружается в системный промпт Claude. Содержит достаточно информации, чтобы Claude понимал, когда следует использовать каждый навык, без загрузки всего содержимого в контекст.

b. **Второй уровень (тело SKILL.md):** загружается, когда Claude считает навык релевантным для текущей задачи. Содержит полные инструкции и рекомендации.

c. **Третий уровень (связанные файлы):** дополнительные файлы, включённые в директорию навыка, которые Claude может просматривать и изучать только по мере необходимости.

Такое постепенное раскрытие **минимизирует использование токенов**, сохраняя специализированную экспертизу.

2. Композиционность

Claude может загружать несколько навыков одновременно. Ваш навык должен хорошо работать в сочетании с другими, не предполагая, что он единственный доступный.

3. Переносимость

Навыки работают идентично на Claude.ai, Claude Code и через API. Создайте навык один раз — он будет работать на всех платформах без изменений, при условии, что среда поддерживает все необходимые зависимости навыка.

Для разработчиков MCP: Навыки и Коннекторы

Создаёте автономные навыки без MCP? Перейдите к разделу «Планирование и дизайн» — вы всегда можете вернуться сюда позже.

Если у вас уже есть работающий MCP-сервер **https://support.claude.com/en/articles/10949351-getting-started-with-local-mcp-servers-on-claude-desktop**, значит, самое сложное вы уже сделали. Навыки — это слой знаний сверху: они фиксируют рабочие процессы и лучшие практики, которые вы уже знаете, чтобы Клод мог применять их последовательно.

Аналогия с кухней

MCP предоставляет профессиональную кухню: доступ к инструментам, ингредиентам и оборудованию.

Навыки — это рецепты: пошаговые инструкции по созданию чего-то ценного.

Вместе они позволяют пользователям выполнять сложные задачи без необходимости разбираться в каждой детали самостоятельно.

Как они работают вместе:

MCP (Связь)	Навыки (Знания)
Подключает Клода к вашему сервису (Notion, Asana, Linear и др.)	Обучает Клода эффективному использованию вашего сервиса
Обеспечивает доступ к данным в реальном времени и вызов инструментов	Фиксирует рабочие процессы и лучшие практики
Что Клод может сделать	**Как** Claude должен это выполнять

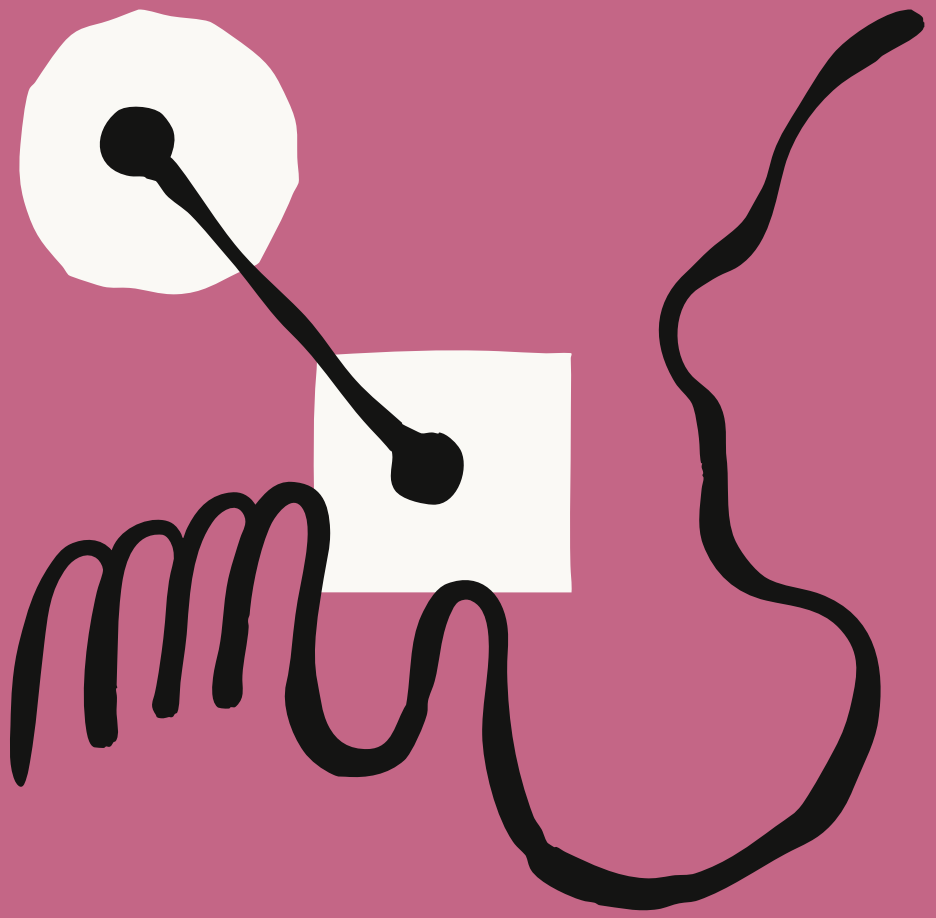
Почему это важно для ваших пользователей MCP

Без навыков:

- Пользователи подключают ваш MCP, но не знают, что делать дальше
- Запросы в службу поддержки с вопросом *«как сделать X с помощью вашей интеграции»*
- Каждый диалог начинается с нуля
- Непоследовательные результаты из-за разных формулировок запросов пользователей
- Пользователи обвиняют ваш коннектор, хотя реальная проблема — в руководстве по рабочему процессу

С навыками:

- Готовые рабочие процессы активируются автоматически при необходимости
- Последовательное и надёжное использование инструментов
- Лучшие практики встроены в каждое взаимодействие
- Сниженный порог вхождения для вашей интеграции



Глава **2**

Планирование и проектирование

Глава **2**

Планирование и проектирование

Начинайте с вариантов использования

Перед написанием кода определите 2–3 конкретных варианта использования, которые должен обеспечивать ваш навык.

Правильное определение варианта использования:

Вариант использования: планирование спринта проекта
Триггер: пользователь говорит «помоги спланировать этот спринт» или «создать задачи для спринта».
Шаги:

1. Получение текущего статуса проекта из Linear (через MCP)
2. Анализ скорости и загрузки команды
3. Предложение приоритетов задач
4. Создание задач в Linear с корректными метками и оценками

Результат: полностью спланированный спринт с созданными задачами

Задайте себе вопросы:

- Чего хочет добиться пользователь?
- Какие многоэтапные рабочие процессы требуются?
- Какие инструменты нужны (встроенные или MCP)?
- Какие отраслевые знания или лучшие практики следует внедрить?

Распространённые категории использования навыков

В Anthropic мы выделили три распространённых сценария использования:

Категория 1: Создание документов и активов

Используется для создания последовательного, высококачественного результата, включая документы, презентации, приложения, дизайны, код и др.

Примеры:

- *навык фронтенд-дизайна https://github.com/anthropics/skills/tree/main/skills/frontend-design*
- *навык для обработки doc https://github.com/anthropics/skills/tree/main/skills/docx*

"Создавайте уникальные frontend-интерфейсы производственного уровня с высоким качеством дизайна. Используйте при создании веб-компонентов, страниц, артефактов, постеров или приложений."

Ключевые техники:

- Встроенные руководства по стилю и стандарты бренда
- Шаблонные структуры для гарантированного результата
- Контрольные списки качества перед финализацией
- Не требует внешних инструментов — использует встроенные возможности Claude

Категория 2: Автоматизация рабочих процессов

Используется для многоэтапных процессов, которые выигрывают от последовательной методологии, включая координацию между несколькими MCP-серверами.

Реальный пример: навык конструктора навыков skill-creator
https://github.com/anthropics/skills/blob/main/skills/skill-creator/SKILL.md

"Интерактивное руководство по созданию новых навыков. Пошагово ведет пользователя через определение сценария использования, генерацию frontmatter (служебной информации), написание инструкций и проверку."

Ключевые техники:

- Пошаговый рабочий процесс с контрольными точками проверки
- Шаблоны для типовых структур
- Встроенный обзор и предложения по улучшению
- Итеративные циклы доработки

Категория 3: Улучшение MCP

Используется для руководства по рабочему процессу с целью расширения доступа к инструментам, предоставляемым MCP-сервером.

Реальный пример: навык sentry-code-review (от Sentry)
https://github.com/getsentry/sentry-for-claude/tree/main/skills

"Автоматически анализирует и исправляет обнаруженные ошибки в GitHub Pull Requests с использованием данных мониторинга ошибок Sentry через их MCP-сервер."

Ключевые техники:

- Координирует несколько последовательных вызовов MCP
- Встраивает экспертные знания в предметной области
- Предоставляет контекст, который пользователям пришлось бы указывать самостоятельно
- Обработка ошибок для типичных проблем MCP

Определите критерии успеха

Как вы узнаете, что ваш навык работает?

Это ориентировочные цели — приблизительные эталоны, а не точные пороговые значения. Стремитесь к строгости, но учитывайте, что всегда присутствует элемент оценки, основанный на интуиции. Мы активно разрабатываем более надёжные рекомендации и инструменты для измерения

Количественные метрики:

- Навык активируется в 90 % соответствующих запросов
  - *Как измерить:* Выполните 10–20 тестовых запросов, которые должны активировать ваш навык. Отслеживайте, сколько раз навык загружается автоматически и сколько раз требует явного вызова.
- Завершает рабочий процесс за X вызовов инструмента
  - *Как измерять:* Сравните выполнение одной и той же задачи с включённым навыком и без него. Подсчитайте количество вызовов инструмента и общее количество потреблённых токенов
- 0 неудачных вызовов API на один рабочий процесс
  - *Как измерять:* Отслеживайте логи MCP-сервера во время тестовых запусков. Отслеживайте частоту повторных попыток и коды ошибок.

Качественные метрики:

- Пользователям не нужно подсказывать Claude, какие шаги выполнять далее
  - *Как оценивать:* Во время тестирования фиксируйте, как часто требуется перенаправление или уточнение. Запрашивайте отзывы у бета-пользователей.
- Рабочие процессы завершаются без исправлений со стороны пользователя
  - *Как оценивать:* Выполните один и тот же запрос 3-5 раз. Сравните результаты на предмет структурной последовательности и качества.
- Последовательные результаты между сессиями
  - *Как оценивать:* Может ли новый пользователь выполнить задачу с первой попытки при минимальных инструкциях?

Технические требования

Структура файлов

```
your-skill-name/
├── SKILL.md                  # Обязательно – основной файл навыка #
├── scripts/                  Необязательно – исполняемый код
│   ├── process_data.py # Пример
│   └── validate.sh # Пример
├── references/               # Необязательно – документация
│   ├── api-guide.md # Пример
│   └── examples/ # Пример
└── assets/                   # Необязательно – шаблоны и прочее
    └── report-template.md # Пример
```

Критические правила

Именование файла SKILL.md:

- Должно быть строго SKILL.md (с учетом регистра)
- Другие варианты не допускаются (SKILL.MD, skill.md и т. п.)

Именование папки навыка:

- Используйте kebab-case: notion-project-setup
- Без пробелов: Notion Project Setup
- Без подчеркиваний: notion_project_setup
- Без заглавных букв: NotionProjectSetup

Ограниченния README.md:

- Не включайте README.md в папку вашего навыка.
- Вся документация должна храниться в SKILL.md или в папке references/
- Примечание: при распространении через GitHub вам всё равно потребуется README на уровне репозитория для пользователей — см. раздел «Распространение и обмен».

YAML-метаданные: самая важная часть

Именно YAML-метаданные определяют, будет ли ваш навык загружен в Claude. Напишите их правильно.

Минимально необходимый формат

```
---
name: your-skill-name
description: Описание функционала. Используется, когда
пользователь запрашивает [конкретные фразы].
---
```

Это всё, что нужно для начала.

Требования к полям

name (обязательно)

- Только kebab-case
- Без пробелов и заглавных букв
- Должно совпадать с именем папки

description (обязательно)

- ДОЛЖНО включать ОБА условия:
  - Что делает навык
  - Когда использовать (условия срабатывания)
- Не более 1024 символов
- Без XML-тегов (< или >)
- Включите конкретные задачи, которые могут озвучивать пользователи
- Укажите типы файлов, если это актуально

license (необязательно)

- Используется при публикации навыка с открытым исходным кодом
- Распространённые: MIT, Apache-2.0

compability (необязательно)

- От 1 до 500 символов
- Указывает требования к окружению: например, целевой продукт, необходимые системные пакеты, требования к сетевому доступу и т.д.

metadata (необязательно):

- Любые пользовательские пары ключ-значение
- Рекомендуемые: автор, версия, mcp-сервер
- *Пример:*

```yaml
metadata:
        author:  ProjectHub
        version: 1.0.0 mcp-server:
        projecthub
```

Ограничения безопасности

Запрещены в frontmatter:

- Угловые скобки XML (< >)
- Навыки с «claude» или «anthropic» в имени (зарезервировано)

Почему: frontmatter появляется в системном запросе Claude.

Злоумышленный контент может внедрять инструкции.

Создание эффективных навыков

Поле описания

Согласно инженерному блогу **https://claude.com/blog/equipping-agents-for-the-real-world-with-agent-skills**. Эти метаданные ... предоставляют достаточно информации, чтобы Claude мог определить, когда следует использовать каждый навык, без необходимости загружать их все в контекст». Это первый уровень поэтапного раскрытия информации.

Структура:

```
[Что делает] + [Когда использовать] + [Ключевые возможности]
```

Примеры хороших описаний:

```
# Хорошее – конкретно и применимо
description: Анализирует дизайнерские файлы Figma и генерирует документацию
для передачи разработчикам. Использовать, когда пользователь загружает
файлы с расширением .fig или запрашивает
«design specs», «component documentation» либо «design-to-code handoff».

# Хорошее – включает триггерные фразы
description: Управляет рабочими процессами проекта Linear, включая
планирование спринтов, создание задач и отслеживание статусов.
Используется, когда пользователь упоминает «спринт», «задачи Linear»,
«планирование проекта» или просит «создать тикеты».

# Хорошее – чёткое ценностное предложение
description: Полный рабочий процесс адаптации клиентов для PayFlow.
Обеспечивает создание аккаунта, настройку платежей и управление подписками.
Используется, когда пользователь говорит
«зарегистрировать нового клиента», «настроить подписку» или
«создать аккаунт PayFlow».
```

Примеры некачественных описаний:

```
# Слишком расплывчато
description: Помогает с проектами.

# Отсутствуют триггеры
description: Создаёт сложные многостраничные системы документации
.

# Слишком техническое, без пользовательских триггеров description:
Реализует модель сущности проекта с иерархиче-
скими связями.
```

Написание основных инструкций

После метаданных напишите основные инструкции в формате Markdown.

Рекомендуемая структура:

Адаптируйте этот шаблон для вашего навыка. Замените выделенные скобками разделы на конкретный контент.

```
---
name: your-skill
description: [--.]
---

# Имя вашего навыка

-# Инструкции
--# Шаг 1: [Первый основной шаг]
Четкое объяснение происходящего.
```

```
Пример:
```bash
python scripts/fetch_data.py --project-id PROJECT_ID
Expected output: [опишите, как выглядит успешное
выполнение]
```

(Добавьте дополнительные шаги при необходимости)

Примеры

Наиболее распространённый сценарий

Пользователь говорит: "Настрой новую маркетинговую кампанию"

Действия:

1. Получить существующие кампании через MCP
2. Создать новую кампанию с заданными параметрами

Результат: Кампания создана с ссылкой для подтверждения

Устранение неполадок

Ошибка: [распространённое сообщение об ошибке]

Причина: [Почему это происходит]

Решение: [Как исправить]

(Добавьте больше случаев ошибок при необходимости)
```

Лучшие практики для инструкций

Будьте конкретны и ориентируйтесь на действия

Хорошо:

Запустите команду **python scripts/validate.py --input {filename}** для проверки формата данных.

Если проверка не пройдена, распространёнными проблемами являются:

--- Отсутствие обязательных полей (добавьте их в CSV)
--- Неверный формат даты (используйте YYYY-MM-DD)

Плохо:

Выполните проверку данных перед продолжением.

Добавьте обработку ошибок

```
-# Распространённые проблемы

--# Ошибка подключения к MCP
Если появляется сообщение «Connection refused»:
1. Убедитесь, что MCP-сервер запущен: проверьте Настройки > Расширения
2. Подтвердите действительность API-ключа
3.  Попробуйте переподключиться: Настройки > Расширения > [Ваш сервис] > Переподключиться
```

Чётко указывайте ссылки на включённые ресурсы

Перед написанием запросов ознакомьтесь с `**references/api-patterns.md**` для следующих аспектов:

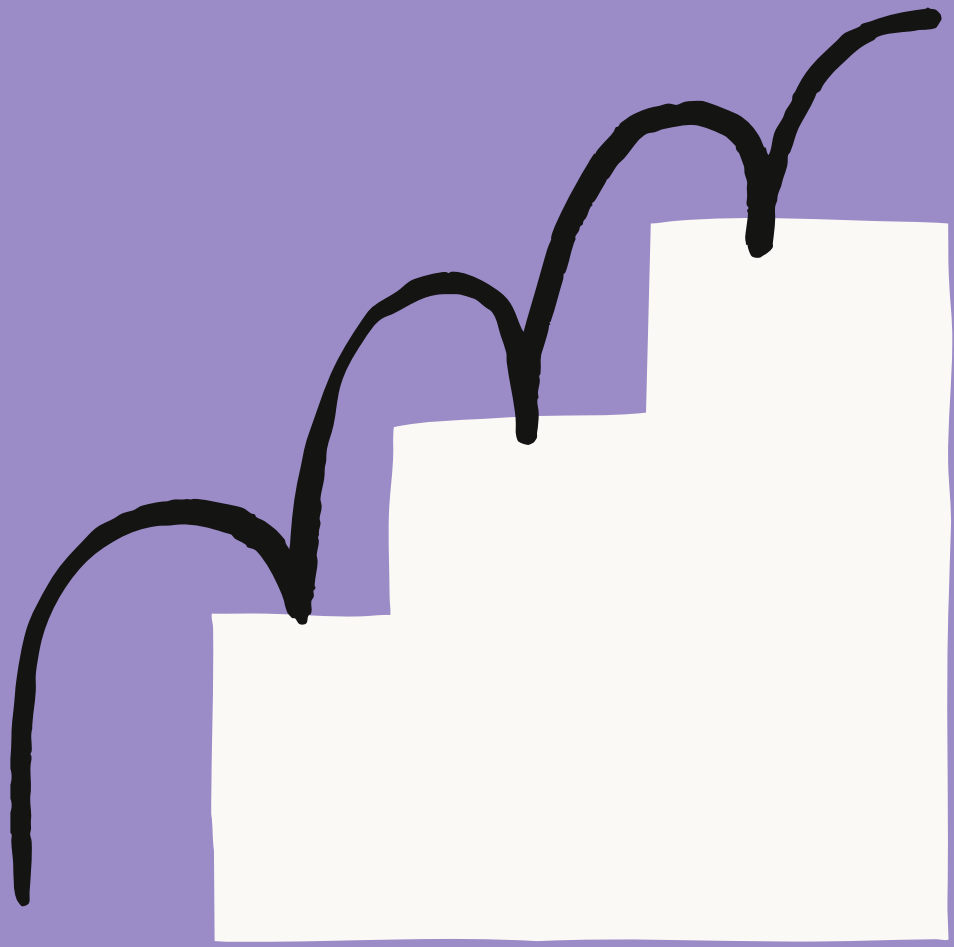
- Рекомендации по ограничению частоты запросов
- Шаблоны пагинации
- Коды ошибок и их обработка

Используйте прогрессивное раскрытие информации

Держите файл SKILL.md сосредоточенным на основных инструкциях.

Перемещайте подробную документацию в ` references/` и делайте на неё ссылки.

(См. раздел "Основные принципы проектирования"для объяснения работы трёхуровневой системы **https://claude.com/blog/equipping-agents-for-the-real-world-with-agent-skills**)



Глава **3**

Тестирование и итерации

Глава 3

Тестирование и итерации

Навыки можно тестировать с разной степенью строгости в зависимости от ваших потребностей:

- **Ручное тестирование в Claude.ai** - Выполняйте запросы напрямую и наблюдайте за поведением. Быстрая итера-ция без необходимости настройки.
- **Скриптовое тестирование в Claude Code** — автоматизируйте тесты для повторимой проверки изменений.
- **Программное тестирование через Skills API** — создавайте наборы тестов, которые выполняются систематически на основе определённых тестовых наборов.

Выберите подход, который соответствует вашим требованиям к качеству и уровню видимости навыка. Навык, используемый внутри небольшой команды, требует другого тестирования, чем навык, развернутый для тысяч корпоративных пользователей.

Полезный совет: Оттачивайте одну задачу до достижения результата перед расширением.

Мы выявили, что самые эффективные создатели навыков оттачивают одну сложную задачу, пока Claude не справится, а затем выделяют успешный подход в отдельный навык. Это использует контекстное обучение Claude и обеспечивает более быстрый отклик, чем широкое тестирование. После того как у вас появится рабочая база, расширяйте тестирование за счёт множества тестовых случаев для полноценного покрытия.

Рекомендуемый подход к тестированию

Основываясь на начальном опыте, эффективное тестирование навыков обычно охватывает три направления:

1. Тесты срабатывания

Цель: Убедиться, что навык активируется в нужный момент.

Кейсы:

- Активируется на очевидных задачах
- Активируется на переформулированных запросах
- Не активируется на нерелевантных темах

Пример набора тестов:

```
Должно активироваться при запросах:
– "Помоги мне настроить новое рабочее пространство ProjectHub"
- "Мне нужно создать проект в ProjectHub"
- "Инициализировать проект ProjectHub для планирования четвертого
квартала"

НЕ должно срабатывать при:
- "Какая погода в Сан-Франциско?"
- "Помоги написать код на Python"
- "Создать таблицу" (если только навык ProjectHub не
  обрабатывает таблицы)
```

2. Функциональные тесты

Цель: Проверить, что навык выдаёт корректные результаты.

Кейсы:

- Получены корректные результаты
- API-вызовы выполнены успешно
- Обработка ошибок работает корректно
- Покрыты граничные случаи

Пример:

```
Тест: Создание проекта с пятью задачами
Дано: имя проекта «Планирование четвертого квартала», 5 описаний
задач Когда: навык выполняет рабочий процесс
Тогда:
  - Проект создан в ProjectHub
  - Созданы 5 задач с правильными свойствами
  - Все задачи связаны с проектом
  - Ошибок API нет
```

3. Сравнение производительности

Цель: доказать, что навык улучшает результаты по сравнению с базовым уровнем.

Используйте метрики определения критериев успеха со страницы 10.

Сравнение с базовым уровнем:

Без навыков

- Пользователь каждый раз предоставляет инструкции
- 15 сообщений туда и обратно
- 3 неудачных вызова API с необходимостью повторной попытки
- Потреблено 12 000 токенов

С навыками

- Автоматическое выполнение рабочего процесса
- Только 2 уточняющих вопроса
- 0 неудачных вызовов API
- Потреблено 6 000 токенов

Использование навыка конструктора навыков

Навык **конструктора навыков** доступен в Claude.ai через каталог плагинов или для скачивания для Claude Code — он поможет создать и доработать навыки. Если у вас есть MCP-сервер и вы знаете 2–3 ключевых рабочих процесса, вы можете создать и протестировать функциональный навык за одно сидение — обычно за 15–30 минут.

Создание навыков:

- Генерация навыков на основе описаний на естественном языке
- Создайте правильно отформатированный файл SKILL.md с фронтматтером
- Предложите триггерные фразы и структуру

Просмотр навыков:

- Отметьте распространённые проблемы (неясные описания, отсутствующие триггеры, структурные ошибки)
- Определите потенциальные риски чрезмерного или недостаточного срабатывания
- Предложите тестовые сценарии на основе заявленной цели навыка

Итеративное улучшение:

- После использования навыка и выявления крайних случаев или ошибок возвращайте эти примеры создателю навыков
- Пример: «Используйте выявленные проблемы и решения в этом чате, чтобы улучшить обработку навыком [конкретного крайнего случая]»

Для использования:

> «Используйте навык **конструктора навыков,** чтобы помочь мне создать навык для [вашего варианта использования]»

Примечание: конструктор навыков помогает проектировать и совершенствовать навыки, но не выполняет автоматизированные тесты и не предоставляет количественные результаты оценки.

Итерация на основе обратной связи

Навыки — это живые документы. Планируйте итерации, учитывая:

Сигналы недостаточного срабатывания:

- Навык не загружается, когда должен
- Пользователи включают его вручную
- Вопросы поддержки относительно случаев использования

> **Решение:** Добавьте больше деталей и нюансов в описание — включая ключевые слова, особенно для технических терминов

Сигналы избыточного срабатывания:

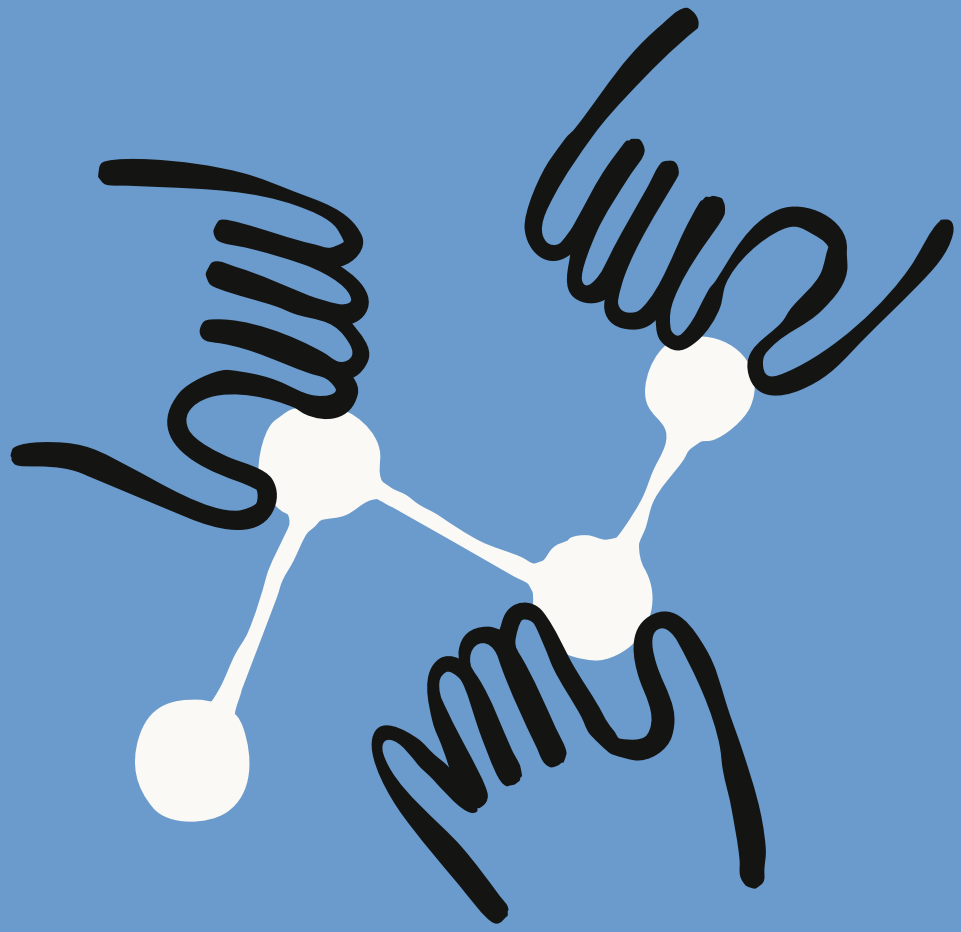
- Загрузка навыка при нерелевантных запросах
- Пользователи отключают его
- Недоразумения по поводу назначения

> Решение: Добавьте отрицательные триггеры, уточните параметры срабатывания

Проблемы с выполнением:

- Непоследовательные результаты
- Сбои вызовов API
- Необходимы исправления со стороны пользователя

> Решение: Улучшите инструкции, добавьте обработку ошибок



Глава **4**

Распространение и обмен

Глава **4**

Распространение и совместное использование

Навыки делают вашу интеграцию MCP более полноценной. При сравнении коннекторов пользователи, выбирающие навыки, получают более быстрый путь к результату, что даёт вам преимущество перед решениями, основанными исключительно на MCP.

Текущая модель распространения (январь 2026 г.)

Как отдельные пользователи получают навыки:

1. Скачать папку навыка
2. Запаковать папку в архив (если необходимо)
3. Загрузить на Claude.ai через Настройки > Возможности > Навыки
4. Или поместить в директорию навыков Claude Code

Навыки на уровне организации:

- Администраторы могут развертывать навыки для всего рабочего пространства (выпущено 18 декабря 2025 года)
- Автоматические обновления
- Централизованное управление

Открытый стандарт

Мы опубликовали https://agentskills.io/home открытый стандарт. Как и в MCP, мы считаем, что навыки должны быть переносимы между инструментами и платформами — один и тот же навык должен работать как в Claude, так и на других AI-платформах. Однако некоторые навыки разработаны для максимального использования возможностей конкретной платформы; авторы могут указать об этом в поле совместимости навыка. Мы сотрудничаем с участниками экосистемы по этому стандарту и рады его раннему принятию.

Использование навыков через API

Для программных сценариев — таких как создание приложений, агентов или автоматизированных рабочих процессов с использованием навыков — API предоставляет прямой контроль над управлением и выполнением навыков.

Ключевые возможности:

- Эндпоинт `/v1/skills` для перечисления и управления навыками
- Добавляйте навыки в запросы Messages API через параметр `container.skills`
- Контроль версий и управление через Claude Console
- Совместимо с Claude Agent SDK для создания пользовательских агентов

Когда использовать навыки через API, а когда через Claude.ai:

Сценарий использования	Оптимальная платформа
Конечные пользователи, взаимодействующие с навыками напрямую	Claude.ai / Claude Code
Ручное тестирование и итерации в процессе разработки	Claude.ai / Claude Code
Индивидуальные, единичные рабочие процессы	Claude.ai / Claude Code
Приложения, использующие навыки программно	API
Масштабное развертывание в производстве	API
Автоматизированные конвейеры и агентские системы	API

Примечание: навыки в API требуют бета-версии Code Execution Tool, обеспечивающей безопаснуюсреду для их выполнения.

Детали реализации:

- **Skills API Quickstart**
https://platform.claude.com/docs/en/agents-and-tools/agent-skills/quickstart
- **Create Custom skills**
https://platform.claude.com/docs/en/api/beta/skills/create
- **Skills in the Agent SDK**
https://platform.claude.com/docs/en/agent-sdk/skills

Рекомендуемый текущий подход

Начните с размещения вашего навыка на GitHub в публичном репозитории с понятным README (для пользователей — отдельно от папки навыка, которая не должна содержать README.md) и примерами использования со скриншотами. Затем добавьте раздел в вашу документацию MCP с ссылкой на навык, объяснением пользы совместного использования и кратким руководством по быстрому старту.

1. Разместите на GitHub
   - Публичный репозиторий для навыков с открытым исходным кодом
   - Понятный README с инструкциями по установке
   - Примеры использования и скриншоты

2. Документируйте в вашем репозитории MCP
   - Ссылки на навыки в документации MCP
   - Объясните ценность совместного использования
   - Предоставьте руководство по быстрому старту

3. Создайте руководство по установке

```
## Installing the [Your Service] skill

1. Download the skill:
- Clone repo: `git clone https://github.com/yourcompany/ skills`
- Or download ZIP from Releases

2. Install in Claude:
- Open Claude.ai > Settings > skills
- Click "Upload skill"
- Select the skill folder (zipped) 3.

Enable the skill:
- Toggle on the [Your Service] skill
- Ensure your MCP server is connected

4. Test:
- Ask Claude: "Set up a new project in [Your Service]"
```

Позиционирование вашего навыка

То, как вы описываете навык, определяет, поймут ли пользователи его ценность и действительно попробуют его. При написании о навыке — в README, документации или маркетинге — придерживайтесь этих принципов.

Сосредоточьтесь на результатах, а не на функциях:

Хорошо:

> «Навык ProjectHub позволяет командам создавать полноценные рабочие пространства проектов за считанные секунды — включая страницы, базы данных и шаблоны — вместо того чтобы тратить 30 минут на ручную настройку.»

Плохо:

> «Навык ProjectHub — это папка, содержащая YAML-метаданные и инструкции в Markdown, которые обращаются к нашим инструментам MCP-сервера.»

Подчеркните взаимодействие MCP и навыков:

> «Наш MCP-сервер предоставляет Claude доступ к вашим проектам в Linear. Наши навыки обучают Claude процессу планирования спринтов вашей команды.» Вместе они обеспечивают управление проектами на базе ИИ.

Глава **5**

Шаблоны и устранение неполадок

Глава **5**

Шаблоны и устранение неполадок

Эти шаблоны возникли из навыков, созданных ранними пользователями и внутренними командами. Они представляют собой распространённые подходы, которые доказали свою эффективность, а не обязательные шаблоны.

Выбор подхода: сначала проблема или сначала инструмент

Представьте, что вы пришли в магазин "Петрович" с проблемой: «Мне нужно починить кухонный шкаф». Сотрудник может указать вам на нужные инструменты, или же вы сами выберете новую дрель и спросите, как использовать ее для конкретной задачи.

Навыки работают так же:

- **Сначала проблема:**
  «Мне нужно создать рабочее пространство для проекта» → Навык организует правильные вызовы MCP в нужной последовательности.

  Пользователи описывают результат; навык управляет инструментами.

- **Сначала инструмент:**
  «У меня подключён Notion MCP» → Навык обучает Claude оптимальным рабочим процессам и лучшим практикам.

  Пользователи имеют доступ; навык предоставляет экспертные знания.

Большинство навыков склоняются в одну из этих сторон. Понимание того, какой фрейминг соответствует вашему случаю использования, помогает выбрать подходящий шаблон ниже.

Шаблон 1: Последовательная оркестрация рабочего процесса

Используется, когда вашим пользователям нужны многоэтапные процессы в конкретном порядке.

Пример структуры:

```
-# Рабочий процесс: Подключение нового клиента

--# Шаг 1: Создание аккаунта
Вызов MCP-инструмента: `create_customer`
Параметры: имя, email, компания

--# Шаг 2: Настройка оплаты
Вызов MCP-инструмента: `setup_payment_method`
Ожидание: подтверждение метода оплаты

--# Шаг 3: Создание подписки
Вызов MCP-инструмента: `create_subscription`
Параметры: plan_id, customer_id (из Шага 1)

--# Шаг 4: Отправка приветственного письма
Вызов MCP-инструмента: `send_email`
Шаблон: welcome_email_template
```

Ключевые техники:

- Явный порядок выполнения шагов
- Зависимости между этапами
- Проверка корректности на каждом этапе
- Инструкции по откату при ошибках

Шаблон 2: Координация нескольких MCP

Используется, когда рабочие процессы охватывают несколько сервисов.

Пример: Передача от дизайна к разработке

```
### Фаза 1: Экспорт дизайна (Figma MCP)
1. Экспортировать дизайн-активы из Figma
2. Сгенерировать технические спецификации дизайна
3. Создать манифест активов

### Фаза 2: Хранение активов (Drive MCP)
1. Создать папку проекта в Drive
2. Загрузить все активы
3. Сформировать ссылки для общего доступа

### Фаза 3: Создание задач (Linear MCP)
1. Создать задачи для разработки
2. Прикрепить ссылки на активы к задачам
3. Назначить задачи инженерной команде

### Фаза 4: Уведомление (Slack MCP)
1. Опубликовать итог передачи в канал #engineering
2. Включить ссылки на ресурсы и ссылки на задачи
```

Ключевые техники:

- Чёткое разделение фаз
- Передача данных между MCP
- Проверка перед переходом к следующей фазе
- Централизованная обработка ошибок

Шаблон 3: Итеративное уточнение

Используется, когда: качество вывода улучшается с итерациями.

Пример: Формирование отчёта

```
## Итеративное создание отчёта

### Начальный черновик
1. Получить данные через MCP
2. Создать начальный черновик отчёта 3.
Сохранить во временный файл

### Контроль качества
1. Запустить скрипт проверки: `scripts/check_report.py`
2. Выявление проблем:
    - Отсутствующие разделы
    - Несоответствие форматирования
    - Ошибки валидации данных

### Цикл доработки
1. Устранение каждой выявленной проблемы
2. Перегенерация затронутых разделов
3. Повторная проверка
4. Повторять до достижения порога качества

### Завершение
1. Применить окончательное форматирование
2. Сгенерировать сводку
3. Сохранить итоговую версию
```

Ключевые техники:

- Четкие критерии качества
- Итеративное улучшение
- Скрипты проверки
- Определить момент завершения итераций

Шаблон 4: Контекстно-зависимый выбор инструмента

Используйте, когда: один и тот же результат достигается разными инструментами в зависимости от контекста.

Пример: хранение файлов

```
## Умное хранение файлов

### Дерево принятия решений
 1. Проверить тип и размер файла
 2. Определить оптимальное место для хранения:
  --- Большие файлы (>10 МБ): использовать облачное хранилище MCP
  --- Совместные документы: использовать Notion/Docs MCP
  --- Файлы кода: использовать GitHub MCP
  --- Временные файлы: использовать локальное хранилище

### Выполнение хранения На
основе решения:
- Вызвать соответствующий MCP-инструмент
- Применить метаданные, специфичные для сервиса
- Сгенерировать ссылку для доступа

### Обеспечить контекст для пользователя
Объяснить, почему выбрано именно это хранилище
```

Ключевые техники:

- Чёткие критерии принятия решения
- Варианты резервных действий
- Прозрачность в отношении выбора

Шаблон 5: Отраслевая интеллектуальная собственность

Используйте, когда: Ваш навык добавляет специализированные знания, выходящие за рамки доступа к инструментам.

Пример: Финансовое соответствие нормативным требованиям

```
## Обработка платежей с соблюдением требований

### Перед обработкой (проверка соответствия)
 1. Получить детали транзакции через MCP
 2. Применить правила соответствия:
    - Проверить санкционные списки
    - Проверить разрешения для юрисдикции
    - Оценить уровень риска
 3. Задокументировать решение по соответствию

### Обработка
ЕСЛИ соответствие подтверждено:
   - Вызвать инструмент обработки платежей MCP
   - Применить соответствующие проверки на мошенничество
   - Обработать транзакцию
В ПРОТИВНОМ СЛУЧАЕ:
   - Отметить для проверки
   - Создать дело о соответствии

### Аудиторский след
- Записывать все проверки соответствия
- Фиксировать решения по обработке
- Создать отчет аудита
```

Ключевые техники:

- Отраслевые знания, встроенные в логику
- Соответствие перед действием
- Полная документация
- Четкое управление

Устранение неполадок

> Навык не загружается

Ошибка: «Не удалось найти SKILL.md в загруженной папке»

Причина: файл не называется точно SKILL.md

Решение:

- Переименуйте в SKILL.md (с учетом регистра)
- Проверьте с помощью: ls -la — должно отображаться SKILL.md

Ошибка: «Недопустимый frontmatter»

Причина: ошибка форматирования YAML

Типичные ошибки:

```
# Wrong - missing delimiters
name: my-skill
description: Does things

# Wrong - unclosed quotes
name: my-skill
description: "Does things

# Correct
---
name: my-skill
description: Does things
---
```

Ошибка: "Недопустимое имя навыка"

Причина: в имени есть пробелы или заглавные буквы

```
# Wrong
name: My Cool Skill

# Correct
name: my-cool-skill
```

> Навык не запускается

Симптом: Навык никогда не загружается автоматически

Решение: Пересмотрите поле описания. См. раздел «Описание» для примеров правильного и неправильного заполнения.

Краткий контрольный список:

- Не слишком ли оно общее? ("Помогает с проектами" не подходит)
- Содержит ли оно триггерные фразы, которые пользователи действительно могут сказать?
- Упоминает ли оно соответствующие типы файлов, если это применимо?

Способ отладки:

Спросите у Claude: «Когда вы бы использовали навык [имя навыка]?» Claude повто-рит описание. Отредактируйте в зависимости от отсутствующих элементов.

> Навык срабатывает слишком часто

Симптом: Навык загружается для нерелевантных запросов

Решения:

1. Добавьте негативные триггеры

```
description: Продвинутый анализ данных для CSV-файлов.
Используйте для статистического моделирования, регрессии и
кластеризации. НЕ используйте для простого исследования данных
(вместо этого используйте навык визуализации данных).
```

2. Будьте более конкретны

```
# Слишком общо
description: Обрабатывает документы

# Более конкретно
description: Обрабатывает PDF-юридические документы для проверки контрактов
```

3. Уточните область применения

```
description: Обработка платежей PayFlow для электронной коммерции. Используйте исключительно для рабочих процессов онлайн-платежей, а не для общих финансовых запросов.
```

> Проблемы с подключением к MCP-серверу

Симптом: Навык загружается, но вызовы MCP-сервера завершаются неудачей

Контрольный список:

1. **Проверьте подключение MCP-сервера**
   - Claude.ai: Настройки > Расширения > [Ваш сервис]
   - Должен отображаться статус «Подключено»
2. **Проверьте аутентификацию**
   - API-ключи действительны и не истекли
   - Выданы надлежащие разрешения/области доступа
   - OAuth-токены обновлены
3. **Проверьте MCP независимо**
   - Попросите Claude вызвать MCP напрямую (без навыка)
   - «Используйте MCP [Сервис] для получения моих проектов»
   - Если это не удаётся, проблема в MCP, а не в навыке
4. **Проверьте имена инструментов**
   - Навык ссылается на правильные имена инструментов MCP
   - Проверьте документацию MCP-сервера
   - Имена инструментов учитывают регистр

> Инструкции не выполнены

Симптом: Навык загружается, но Claude не выполняет инструкции

Распространённые причины:

1. Инструкции слишком многословны
   - Держите инструкции лаконичными
   - Используйте маркированные и нумерованные списки
   - Переместите подробную справочную информацию в отдельные файлы
2. Инструкции спрятаны
   - Разместите критически важные инструкции в начале
   - Используйте заголовки **## Важно** или **## Критично**
   - При необходимости повторяйте ключевые моменты
3. Неоднозначный язык

```
# Плохо
 Обязательно корректно проверяйте данные

 # Хорошо
 КРИТИЧНО: Перед вызовом create_project убедитесь в следующем:
 - Имя проекта не пустое
 - Назначен хотя бы один участник команды
 - Дата начала не в прошлом
```

Продвинутый метод: для критических проверок рассмотрите интеграцию скрипта, который программно выполняет проверки, вместо опоры на языковую интерпретацию. Код детерминирован, но интерпретация языка — нет.

Примеры такого подхода - в навыках Office: **https://github.com/anthropics/skills/tree/main/skills.**

4. "Ленивая модель": добавьте явное стимулирование

```
# Примечания по производительности
- Не торопитесь, выполняйте это тщательно.
- Качество важнее скорости.
- Не пропускайте этапы проверки.
```

Примечание: добавление такой модели в **промпт пользователя** эффективнее, чем в SKILL.md.

Проблемы с большим контекстом.

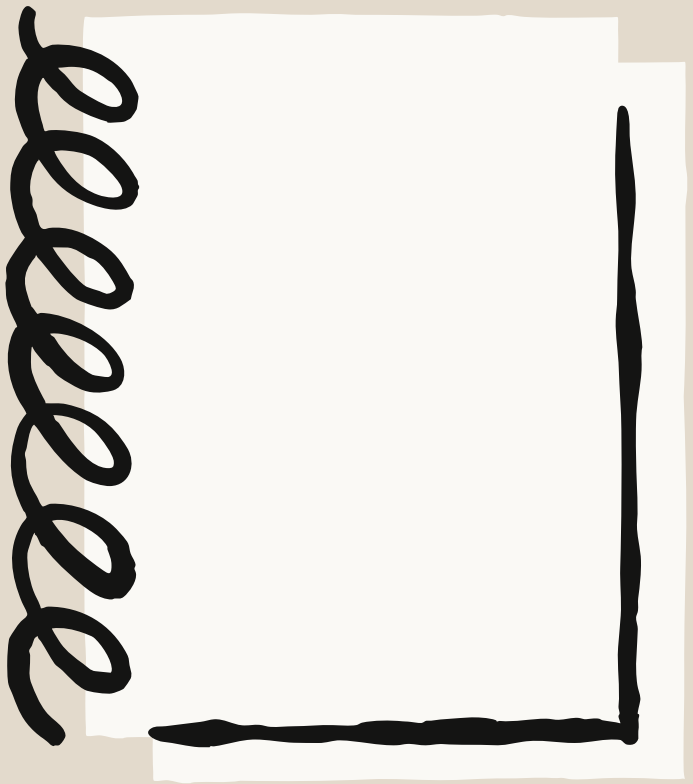
Симптом: Навык работает медленно или ответы ухудшились.

Причины:

- Содержимое навыка слишком велико.
- Одновременное включение слишком большого количества навыков.
- Загружено всё содержимое вместо постепенного раскрытия.

Решения:

1. Оптимизируйте размер SKILL.md
   - Перенесите подробную документацию в папку references/
   - Ставьте ссылки на references вместо вставки текста
   - Содержимое SKILL.md должно быть менее 5 000 слов.
2. Сократите количество включённых навыков.
   - Проверьте, если у вас одновременно включено более 20–50 навыков.
   - Рекомендуется избирательное включение.
   - Рассмотрите возможность использования "упаковок" навыков для связанных функций



Глава **6**

Ресурсы и ссылки

Глава **6**

Ресурсы и ссылки

Если вы создаёте свой первый навык, начните с **Руководства по лучшим практикам,** затем при необходимости обращайтесь к документации API.

Официальная документация

Ресурсы Anthropic:

Best Practices Guide
https://platform.claude.com/docs/en/**agents-and-tools/agent-skills/best-practices**

Skills Documentation
https://platform.claude.com/docs/en/**agents-and-tools/agent-skills/overview**

API Reference
https://platform.claude.com/docs/en/**api/overview**

MCP Documentation
https://**modelcontextprotocol**.io/docs/getting-started/intro

Публикации в блогах:

Введение в навыки агентов
https://claude.com/blog/skills

Инженерный блог: оснащение агентов для реального мира
https://claude.com/blog/equipping-agents-for-the-real-world-with-agent-skills

Объяснение навыков
https://claude.com/blog/skills-explained

Как создавать навыки для Claude
https://web.archive.org/web/20260123153945/https://claude.com/blog/how-to-create-skills-key-steps-limitations-and-examples

Создание навыков для Claude Code
https://claude.com/blog/building-skills-for-claude-code

Улучшение дизайна фронтенда с помощью навыков
https://claude.com/blog/improving-frontend-design-through-skills

Примеры навыков

Публичное репозиторий навыков:

https://github.com/anthropics/skills

Содержит навыки, созданные Anthropic, которые можно настроить

Инструменты и утилиты

Навык **skill-creator**:

- Встроен в Claude.ai и доступен для Claude Code
- Может создавать навыки на основе описаний
- Проводит обзор и предоставляет рекомендации
- Использование: «Помогите мне создать навык с помощью создателя навыков»

Валидация:

- Skill-creator может оценить ваши навыки
- Спросите его: «Проверьте этот навык и предложите улучшения»

Получение поддержки

По техническим вопросам:

Общие вопросы: форумы сообщества в Claude Developers Discord
https://discord.com/invite/6PPFFzqPDZ

Для сообщений об ошибках:

- GitHub Issues: https://github.com/anthropics/skills/issues
- Используйте: имя навыка, сообщение об ошибке, шаги для воспроизведения

Справка A: быстрый контрольный список

Используйте этот контрольный список для проверки навыка до и после загрузки. Если хотите начать быстрее, используйте навык **skill_creator** для создания первого черновика, а затем пройдитесь по этому списку, чтобы убедиться, что ничего не пропустили.

Перед началом работы

- [ ] Определены 2–3 конкретных сценария использования
- [ ] Определены инструменты (встроенные или MCP)
- [ ] Изучено это руководство и пример навыков
- [ ] Спланирована структура папок

В процессе разработки

- [ ] Папка названа в формате kebab-case
- [ ] Файл SKILL.md существует (точное написание)
- [ ] YAML-метаданные имеют разделители ---
- [ ] Поле имя: kebab-case, без пробелов и заглавных букв
- [ ] Описание включает ЧТО и КОГДА
- [ ] Теги XML (< >) нигде не используются
- [ ] Инструкции ясные и практические
- [ ] Обработка ошибок предусмотрена
- [ ] Предоставлены примеры
- [ ] Ссылки чётко указаны

Перед загрузкой

- [ ] Проверено срабатывание на очевидных задачах
- [ ] Проверено срабатывание на перефразированных запросах
- [ ] Подтверждено отсутствие срабатывания на нерелевантных темах
- [ ] Функциональные тесты пройдены
- [ ] Интеграция с инструментами работает (если применимо)
- [ ] Сжатие в файл формата .zip

После загрузки

- [ ] Тестирование в реальных беседах
- [ ] Мониторинг на предмет недосрабатывания или избыточного срабатывания
- [ ] Сбор обратной связи от пользователей
- [ ] Итеративное улучшение описания и инструкций
- [ ] Обновление версии в метаданных

Справка B: YAML-метаданные

Обязательные поля

```
---
name: skill-name-in-kebab-case
description: Что делает навык и когда его использовать. Включите конкретные
триггерные фразы.
---
```

Все необязательные поля

```
name: skill-name
description: [обязательное описание]
license: MIT # Необязательно: лицензия для open-source
allowed-tools: "Bash(python:*) Bash(npm:*) WebFetch" # Необязательно: Ограничить доступ к инструментам
metadata: # Необязательно: пользовательские поля
      автор: Название компании
      версия: 1.0.0
      mcp-сервер: server-name
      категория: продуктивность
      теги: [управление-проектами, автоматизация]
      документация: https://example.com/docs
      техническая поддержка: support@example.com
```

Заметки по безопасности

Разрешено:

- Любые стандартные типы YAML (строки, числа, логические значения, списки, объекты)
- Пользовательские поля метаданных
- Подробные описания (до 1024 символов)

Запрещено:

- Угловые скобки XML (< >) – ограничение безопасности
- Выполнение кода в YAML (используется безопасный парсер YAML)
- Навыки с именами, начинающимися с «claude» или «anthropic» (зарезервировано)

Справка C: Полные примеры навыков

Для полноценных навыков, готовых к производству и демонстрирующих шаблоны из этого руководства:

- Навыки работы с документами
  `https://github.com/anthropics/skills/tree/main/skills/pdf`
  `https://github.com/anthropics/skills/tree/main/skills/docx`
  `https://github.com/anthropics/skills/tree/main/skills/pptx`
  `https://github.com/anthropics/skills/tree/main/skills/xlsx`

- Различные шаблоны рабочих процессов
  `https://github.com/anthropics/skills/tree/main/skills`

- Каталог таких партнёров как Asana, Atlassian, Canva, Figma, Sentry, Zapier и другие
  `https://claude.com/connectors`

Эти репозитории регулярно обновляются и содержат дополнительные примеры, выходящие за рамки изложенного здесь.

Клонируйте их, модифи-цируйте под свои задачи и используйте как шаблоны.

Claude

claude.ai