Скиллы в Клод Код с нуля

Полный путь от идеи до рабочего скилла. Без знания кода

Что вы получите из этого гайда

- Понимание, **что такое скилл** и чем он отличается от инструкций в `CLAUDE.md`.
- Структуру папки скилла: `SKILL.md`, `scripts/`, `references/` - что куда класть.
- Как **описание скилла** триггерит автоматическую загрузку Claude и почему это самое важное поле.
- Когда выносить логику в скрипт, а когда оставить инструкцией прямо в `SKILL.md`.
- Где живут скиллы: личные (`~/.claude/skills/`) против проектных (`.claude/skills/`).
- Разбор одного реального рабочего скилла пошагово.

Время чтения: ≈ 13 минут. **Уровень:** для тех, кто уже работает с Claude Code и понимает, что такое `CLAUDE.md`.

> Если ещё не пользовались Claude Code - сначала возьмите «Клод Код за вечер», там все базовые шаги. Потом возвращайтесь сюда.

Что такое скилл

Скилл - это **миниавтоматизация, которую вы создаёте один раз и используете снова и снова**. Personal AI-помощник под конкретную задачу: обработка счетов, разбор почты, отправка отчёта клиенту, проверка чек-листа перед публикацией.

Два примера из моей повседневной работы:

Gmail digest. Скилл анализирует почту за заданный период и раскладывает на три группы: требуют ответа, средний приоритет (хорошо бы прочесть), неважные (помечаются прочитанными автоматически). Утренние 100 писем превращаются в 30-секундный обзор.

Invoice extraction. Скилл берёт счёт-фактуру (PDF или фото), вытягивает ключевые поля, складывает в Google Sheets. На основе накопленных данных может построить месячный отчёт с аналитикой по поставщикам и затратам.

Любой повторяющийся процесс - кандидат на скилл.

Скилл против `CLAUDE.md`: в чём разница

`CLAUDE.md` - **читается при каждой новой сессии**. Туда идёт то, что должно быть всегда: общие правила работы, контекст проекта, индекс других файлов.

Скилл - **загружается только когда триггерится**. Сидит в папке как файл, не съедает токены пока не нужен. Когда вы или Claude вызывает его - содержимое подгружается в контекст одним блоком.

Если инструкцию нужно применять «всегда» - в `CLAUDE.md`. Если «иногда, под конкретный сценарий» - в скилл.

Шаг 1. Структура папки скилла (≈ 2 минуты)

Каждый скилл - это отдельная папка. Внутри минимум один файл:

```
my-skill/
├── SKILL.md           ← обязательно (главный файл с инструкциями)
├── scripts/
│   └── helper.py      ← опционально (исполняемая логика)
├── references/
│   └── examples.md    ← опционально (справочные материалы)
└── examples/
    └── sample.md      ← опционально (примеры ожидаемого результата)
```

`SKILL.md` - единственный обязательный. Там два блока:

1. **YAML frontmatter** между `---` и `---` - метаданные (имя, описание, разрешённые инструменты).
2. **Markdown-инструкция** - что Claude делает, когда скилл активируется.

`scripts/` - всё, что должно реально *выполняться*: Python-скрипты, bash-команды, обработка файлов. Claude запускает их, не читает построчно.

`references/` - справочные материалы, шаблоны, длинные примеры. Загружаются только когда скилл прямо ссылается на них из `SKILL.md`. Не съедают контекст пока не нужны.

`examples/` - примеры ожидаемого результата. Полезно для скиллов, которые что-то генерируют (Claude видит образец и старается ему соответствовать).

Шаг 2. YAML frontmatter (≈ 3 минуты)

Frontmatter - это блок метаданных в самом верху `SKILL.md`. Минимальный пример:

```
---
name: summarize-changes
description: Кратко описывает несохранённые изменения в репозитории и помечает
рискованные. Использовать когда пользователь спрашивает «что я изменил» или просит
сообщение для коммита.
---
```

Все поля **опциональны**, но `description` - **самое важное**. На него ориентируется Claude, когда решает, какой скилл использовать в текущей задаче.

Ключевые поля

ПОЛЕ	ЗАЧЕМ	КОГДА ИСПОЛЬЗОВАТЬ
`name`	Имя скилла, превращается в slash-команду `/skill-name`	Если хотите имя отличное от названия папки
`description`	**Триггер для автоматической загрузки**. Claude читает у всех скиллов и выбирает подходящий	Всегда. Без этого скилл «висит» и не вызывается автоматически
`disable-model-invocation`	`true` запрещает Claude вызывать скилл автоматически - только вручную через `/skill-name`	Для скиллов с побочными эффектами: деплой, отправка письма, операции с деньгами
`allowed-tools`	Список инструментов, которые скилл может использовать без отдельного подтверждения	Когда нужно, чтобы Claude мог запустить, например, `Bash(git *)` без вопросов
`argument-hint`	Подсказка для автозаполнения, что передавать в скилл	`[issue-number]`, `[filename]` - читателю видно при вводе `/skill-name`

Описание - ключ к автоматическому триггеру

`description` - это не просто «что делает скилл». Это **инструкция для Claude когда его вызывать**. Поэтому туда стоит написать:

- Что делает (одной фразой).
- Когда использовать (триггерные фразы).
- Когда **не использовать** (negative triggers - чтобы скилл не срабатывал не вовремя).

Хороший пример описания (из реального скилла):

> *Search YouTube by topic and return structured results with views, subscribers, duration. Triggers on: «search youtube», «find youtube videos about», «trending videos on», «youtube research», «/youtube-search». Used by other skills as a content discovery source. Does NOT trigger for fetching a specific channel's uploads (use tool-youtube channel mode) or pulling transcripts from a known URL.*

Здесь видно всё: что делает, на каких фразах срабатывает, **на каких не должен срабатывать** (negative triggers), и связь с другими скиллами в системе.

> **Важная деталь.** Описание + when_to_use ограничены 1536 символами в системном листинге. Самое важное - в начало.

Шаг 3. Когда выносить в скрипт vs инструкцией (≈ 2 минуты)

Распространённая ошибка - писать в `SKILL.md` детальные шаги, которые на самом деле должен выполнять код.

Оставляйте инструкцией в `SKILL.md` когда: - Нужно решение в стиле «Claude посмотри на это и реши». - Шаги зависят от контекста и не одинаковы каждый раз. - Это про *понимание* и *интерпретацию*: проверь стиль текста, оцени качество, переведи с учётом тона.

Выносите в `scripts/` когда: - Это всегда одни и те же шаги без вариативности (поиск файла, вызов API, обработка данных). - Нужны библиотеки и зависимости (pandas, requests, yt-dlp). - Логика должна работать одинаково *каждый раз*.

Связка работает так: `SKILL.md` объясняет «что» и «зачем», скрипт делает «как». Claude запускает скрипт и работает с его выходом.

Пример из `tool-youtube-search`: `SKILL.md` объясняет логику поиска и его связь с другими скиллами, а вся реальная работа с YouTube идёт через `scripts/search.py` (вызов `yt-dlp` с нужными параметрами).

Шаг 4. Где живут скиллы (≈ 1 минута)

Расположение определяет, кто может использовать скилл:

ГДЕ ЛЕЖИТ	ПУТЬ	ДОСТУПЕН
Личный	`~/.claude/skills/<имя>/SKILL.md`	Во всех ваших проектах
Проектный	`.claude/skills/<имя>/SKILL.md`	Только в этом проекте
Плагин	`<plugin>/skills/<имя>/SKILL.md`	Где плагин подключён

Личные скиллы - универсальные помощники. Например, скилл для составления коммит-сообщений: одинаково полезен в любом репозитории. Кладёте в `~/.claude/skills/`.

Проектные скиллы - специфичны для одного бизнеса/клиента/проекта. Например, скилл «обработай инвойс по нашему шаблону» - имеет смысл только внутри проекта определённой компании. Кладёте в `.claude/skills/` внутри папки проекта.

Если скиллы с одним именем есть и там, и там - **проектный имеет приоритет** для текущего проекта.

Шаг 5. Разбор реального скилла (≈ 3 минуты)

Возьмём `tool-youtube-search` - простой рабочий скилл, который ищет видео на YouTube по теме. Покажу, как он устроен.

Frontmatter

```
---
name: tool-youtube-search
description: Search YouTube by topic and return structured results with views,
  subscribers, duration, upload date, and views/subs ratio. Powered by yt-dlp -
  no Google API key required. Use to find trending videos in a niche, build a
  research corpus, or scout competitors. Triggers on: "search youtube", "find
  youtube videos about", "trending videos on", "youtube research",
  "/youtube-search". Used by other skills (mkt-youtube-script, viz-instagram-
  carousel) as a content discovery source. Does NOT trigger for fetching a
  specific channel's uploads or pulling transcripts from a known URL.
---
```

Что важно: - **Имя** совпадает с папкой - так появляется команда `/youtube-search`. - **Описание** жирно расписывает триггеры и анти-триггеры. Claude теперь точно знает, когда вызывать. - **Связи** упомянуты: «used by mkt-youtube-script, viz-instagram-carousel» - так в системе видны зависимости.

Тело SKILL.md

В теле кратко описано, что делает скилл, какие зависимости (yt-dlp), и куда обратиться если что-то не работает. Главное - **ссылка на скрипт**:

```
python .claude/skills/tool-youtube-search/scripts/search.py "claude code skills" --
count 20 --months 1
```

Тут в одной строке вся логика для Claude: запусти этот скрипт с такими аргументами, он сам сделает поиск и вернёт результат.

Скрипт

Реальная работа с yt-dlp в `scripts/search.py`. Скилл-инструкция (`SKILL.md`) не пытается описать «как сделать запрос к YouTube» - это работа кода. `SKILL.md` объясняет *когда* и *как использовать*, скрипт делает *что*.

Тестирование скилла

После того как папка с `SKILL.md` создана, скилл загружается **в текущей сессии Claude Code** автоматически - перезапускать ничего не нужно.

Проверить можно двумя способами:

1. **Дать Claude фразу из триггеров:** *«Найди трендовые видео про Claude Code за последний месяц»* - Claude должен сам подтянуть скилл.
2. **Вызвать вручную:** `/youtube-search claude code skills` - срабатывает напрямую.

Если Claude не подхватывает скилл автоматически - усильте описание: добавьте больше триггерных фраз, конкретных формулировок которые пользователь может сказать.

Частые ошибки новичков

Скилл не вызывается автоматически. Слабое описание. Добавьте конкретные фразы которые пользователь может сказать. Не «work with files» - а *«rename files in batch», «process invoices», «sort folder by date»*.

Скилл срабатывает не вовремя. Слишком общее описание. Сузьте: добавьте «**Does NOT trigger** for...» - явно укажите когда не использовать.

В `SKILL.md` много кода. Вынесите в `scripts/`. `SKILL.md` - это инструкция для Claude, не место для логики. 500 строк - максимум, в идеале меньше.

Скилл забит примерами и справочниками. Вынесите в `references/`. Эти файлы загружаются только когда `SKILL.md` на них ссылается. Иначе раздуваете контекст без пользы.

Личный или проектный? Если скилл универсальный (работает с любым репо/проектом) - в `~/.claude/skills/`. Если специфичный для одного клиента/бизнеса - в `.claude/skills/` проекта.

После этого гайда

Когда первый скилл собран и работает - возвращайтесь в каталог за следующими материалами:

- **«17 приёмов в Клод Код»** - конкретные приёмы из ежедневной работы, включая правила для `CLAUDE.md`, которые усиливают взаимодействие со скиллами.
- **«98% токенов впустую»** - 16 советов как не сжигать контекст. Особенно актуально, когда скиллов становится много.
- **«Клод Код + Notebook LM»** - связка двух инструментов для исследований и контента.

Подписывайтесь, чтобы видеть новое

YouTube · **@ai_with_marina** Подробные разборы AI-инструментов и пошаговые гайды для русскоязычной аудитории. Новое видео раз в неделю.

Telegram · **@ai_with_marina** Короткие заметки, анонсы новых материалов AIAtlas, обсуждения и быстрые ответы на вопросы.

Если что-то в гайде не работает или нашли неточность - напишите в Telegram-канал, отвечу.