Как установить 9 Claude Skills: пошаговая инструкция

Skill 1: UI/UX: frontend-design

Что делает скилл:

превращает Claude в дизайнера. Активируется автоматически при UI-задачах.

GitHub: https://github.com/anthropics/claude-code/tree/main/plugins/frontend-design

Способ 1 — через npx skills (рекомендуется)

```
npx skills add anthropics/claude-code skill frontend-design
```

Требования: Node.js 18+, npm/npx доступны в терминале.

Способ 2 — через Claude Code slash-команду

Открой Claude Code и выполни:

```
/plugin install frontend-design@claude-plugins-official
```

Способ 3 — ручная установка (fallback)

```
git clone --depth 1 https://github.com/anthropics/claude-code.git /tmp/anthropics-claude-code
```

```
find /tmp/anthropics-claude-code -type d -iname "frontend-design"
```

```
cp -r /tmp/anthropics-claude-code/plugins/frontend-design ~/.claude/skills/frontend-design
```

1. Клонируй репозиторий:
2. Найди папку скилла:
3. Скопируй в директорию скиллов:

Проверка установки

```
ls ~/.claude/skills/frontend-design/SKILL.md
```

Если файл существует — скилл установлен.

Skill 2: web-design-guidelines

GitHub: https://github.com/vercel-labs/agent-skills/blob/main/skills/web-design-guidelines/SKILL.md

Что делает скилл

Аудитирует существующий UI по 100+ правилам. Выдаёт список проблем в формате `file:line`.

Важно: идёт в комплекте с `react-best-practices` — одна команда устанавливает оба скилла.

Способ 1 — через npx skills (рекомендуется)

```
npx skills add vercel-labs/agent-skills
```

Эта команда устанавливает весь пакет Vercel Labs, включая `web-design-guidelines` и `react-best-practices` одновременно.

Способ 2 — ручная установка (fallback)

1. Клонируй репозиторий:

```
git clone --depth 1 https://github.com/vercel-labs/agent-skills.git /tmp/vercel-agent-skills
```

1. Скопируй нужный скилл:

```
cp -r /tmp/vercel-agent-skills/skills/web-design-guidelines ~/.claude/skills/web-design-guidelines
```

Проверка установки

```
ls ~/.claude/skills/web-design-guidelines/SKILL.md
```

Skill 3: shadcn-ui

Docs: https://ui.shadcn.com/docs/skills

Changelog: https://ui.shadcn.com/docs/changelog/2026-03-cli-v4

Что делает скилл

Project-aware: читает `components.json`, понимает фреймворк, алиасы, установленные компоненты и иконочную библиотеку. Генерит код с правильными импортами, `FieldGroup`, `data-invalid` — сразу работает без правок.

Требование: проект должен быть инициализирован через `npx shadcn init` (наличие `components.json`).

Способ 1 — через shadcn CLI (рекомендуется)

Убедись, что в проекте есть `components.json` (стандартный файл shadcn-проекта):

```
# Если shadcn уже инициализирован в проекте:
npx shadcn skills add shadcn-ui
```

Скилл автоматически читает `components.json` и адаптируется под твой проект.

Способ 2 — через npx skills

```
npx skills add shadcn-ui/ui skill shadcn-ui
```

Способ 3 — ручная установка (fallback)

1. Скачай SKILL.md напрямую:

```
mkdir -p ~/.claude/skills/shadcn-ui
curl -L https://ui.shadcn.com/docs/skills/SKILL.md -o ~/.claude/skills/shadcn-ui/SKILL.md
```

1. Убедись, что `components.json` находится в корне проекта — скилл его читает при активации.

Проверка установки

```
ls ~/.claude/skills/shadcn-ui/SKILL.md
cat <project-root>/components.json  # должен существовать
```

Skill 4: superpowers

GitHub: https://github.com/obra/superpowers

Marketplace: https://github.com/obra/superpowers-marketplace

Что делает скилл

Фреймворк из 20+ скиллов. `/brainstorm` → уточняет спеку. `/write-plan` → разбивает на задачи по 2-5 мин с тестами. `/execute-plan` → кодит с обязательным TDD (красный → зелёный → рефактор). Запрещает писать «тест допишу потом».

Способ 1 — через Claude plugin page (рекомендуется)

Перейди на страницу плагина и нажми Install:

https://claude.com/plugins/superpowers

Способ 2 — через npx skills

```
npx skills add obra/superpowers
```

Способ 3 — через superpowers-skills репозиторий

```
npx skills add obra/superpowers-skills
```

Способ 4 — ручная установка (fallback)

1. Клонируй репозиторий:

```
git clone --depth 1 https://github.com/obra/superpowers.git /tmp/superpowers
git clone --depth 1 https://github.com/obra/superpowers-skills.git /tmp/superpowers-skills
```

1. Скопируй скиллы:

```
cp -r /tmp/superpowers-skills/skills/* ~/.claude/skills/
```

Проверка установки

После установки в Claude Code должны работать команды:

```
/brainstorm
/write-plan
/execute-plan
```

Skill 5: react-best-practices

GitHub: https://github.com/vercel-labs/agent-skills/blob/main/skills/react-best-practices/SKILL.md

Blog: https://vercel.com/blog/introducing-react-best-practices

Что делает скилл

Проходится по React/Next.js коду и находит анти-паттерны: лишние `use client`, `useEffect` вместо Server Component, `key={index}` в списках, лишние клиентские бандлы. Выдаёт конкретные рекомендации с указанием файла и строки.

8 категорий правил:

- Server Components
- Client Components
- Data Fetching
- Caching
- Routing
- Performance
- Accessibility
- TypeScript

Способ 1 — через npx skills (рекомендуется)

```
npx skills add vercel-labs/agent-skills
```

Важно: эта команда устанавливает весь пакет Vercel Labs. Скиллы `react-best-practices` и `web-design-guidelines` устанавливаются одновременно одной командой.

Способ 2 — ручная установка (fallback)

1. Клонируй репозиторий (если ещё не клонировал для web-design-guidelines):

```
git clone --depth 1 https://github.com/vercel-labs/agent-skills.git /tmp/vercel-agent-skills
```

1. Скопируй скилл:

```
cp -r /tmp/vercel-agent-skills/skills/react-best-practices ~/.claude/skills/react-best-practices
```

Проверка установки

```
ls ~/.claude/skills/react-best-practices/SKILL.md
```

Skill 6: postgres-best-practices

Listing: https://mcpservers.org/agent-skills/supabase/postgres-best-practices

Альтернатива: https://claudemarketplaces.com/skills/planetscale/database-skills/postgres

Что делает скилл

Анализирует SQL-запросы и схему базы. Находит: отсутствующие индексы, неоптимальные запросы, небезопасные миграции. Даёт рекомендации по: performance, zero-downtime migrations, Row Level Security (RLS), materialized views, partial indexes.

> **Реальный результат:** 4 отсутствующих индекса → время критического запроса с 3 сек до 80 мс.

Способ 1 — через npx skills (рекомендуется)

```
npx skills add supabase/postgres-best-practices
```

Способ 2 — через Supabase CLI

Если у тебя установлен Supabase CLI:

```
supabase skills add postgres-best-practices
```

Способ 3 — ручная установка (fallback)

1. Клонируй репозиторий Supabase skills:

```
git clone --depth 1 https://github.com/supabase/agent-skills.git /tmp/supabase-skills
```

1. Найди и скопируй скилл:

```
find /tmp/supabase-skills -type d -iname "postgres*"
cp -r /tmp/supabase-skills/skills/postgres-best-practices ~/.claude/skills/postgres-best-practices
```

1. Если репозиторий недоступен — скачай SKILL.md напрямую со страницы листинга и сохрани в `~/.claude/skills/postgres-best-practices/SKILL.md`.

Проверка установки

```
ls ~/.claude/skills/postgres-best-practices/SKILL.md
```

Skill 7: mcp-builder

GitHub: https://github.com/anthropics/skills

Docs: https://resources.anthropic.com/hubfs/The-Complete-Guide-to-Building-Skill-for-Claude.pdf

Что делает скилл

Проектирует полноценные MCP-серверы (Python или TypeScript) по описанию задачи. Включает: дизайн инструментов, типизацию, обработку ошибок, тесты. Позволяет подключить Claude к любому внутреннему API за одну сессию.

> **Пример:** MCP для YouTube-аналитики из CSV → Notion за 40 минут без бойлерплейта.

Способ 1 — через npx skills (рекомендуется)

```
npx skills add anthropics/skills skill mcp-builder
```

Способ 2 — через Claude Code slash-команду

```
/plugin install mcp-builder@claude-plugins-official
```

Способ 3 — ручная установка (fallback)

1. Клонируй официальный репозиторий Anthropic skills:

```
git clone --depth 1 https://github.com/anthropics/skills.git /tmp/anthropics-skills
```

1. Найди папку mcp-builder:

```
find /tmp/anthropics-skills -type d -iname "mcp*"
```

1. Скопируй в директорию скиллов:

```
cp -r /tmp/anthropics-skills/mcp-builder ~/.claude/skills/mcp-builder
```

Проверка установки

```
ls ~/.claude/skills/mcp-builder/SKILL.md
```

Skill 8: canvas-design

GitHub: https://github.com/ComposioHQ/awesome-claude-skills/blob/master/canvas-design/SKILL.md

Связанный репо: https://github.com/anthropics/skills

Что делает скилл

Генерирует PNG и PDF через код (не через диффузию). Постеры, обложки, инфографики, социальные баннеры. Результат параметризован: шрифт, цвет, расположение блоков — всё правится.

> **Реальный результат:** обложки уроков — с 30 мин в Figma до 3 мин на 5 штук.

Способ 1 — через npx skills (рекомендуется)

```
npx skills add anthropics/skills skill canvas-design
```

Способ 2 — через ComposioHQ репозиторий

```
npx skills add ComposioHQ/awesome-claude-skills skill canvas-design
```

Способ 3 — ручная установка (fallback)

1. Скачай SKILL.md напрямую:

```
mkdir -p ~/.claude/skills/canvas-design
curl -L "https://raw.githubusercontent.com/ComposioHQ/awesome-claude-skills/master/canvas-design/SKILL.md" \
  -o ~/.claude/skills/canvas-design/SKILL.md
```

1. Если нужны дополнительные ассеты — клонируй репозиторий:

```
git clone --depth 1 https://github.com/ComposioHQ/awesome-claude-skills.git /tmp/composio-skills
cp -r /tmp/composio-skills/canvas-design ~/.claude/skills/canvas-design
```

Проверка установки

```
ls ~/.claude/skills/canvas-design/SKILL.md
```

Skill 9: agent-skills CLI

GitHub: https://github.com/vercel-labs/agent-skills

CLI npm: https://github.com/vercel-labs/skills

Docs: https://vercel.com/docs/agent-resources/skills

Что делает скилл

Пакетный менеджер для скиллов. `npx skills add <org>/<repo>` устанавливает любой скилл в Claude Code, Cursor, Codex, Gemini CLI, OpenCode и 50+ агентов одновременно. Один формат — вся экосистема.

Главная фича: список скиллов в `package.json` → `npm install` → все в команде получают одинаковый стек агентских инструментов автоматически.

Способ 1 — установка самого CLI (это и есть скилл)

```
npm install -g @vercel/skills
# или без глобальной установки:
npx @vercel/skills --help
```

Способ 2 — добавить в проект как dev-зависимость

```
npm install --save-dev @vercel/skills
```

Затем добавь в `package.json`:

```
{
 "scripts": {
  "skills:install": "npx skills add vercel-labs/agent-skills && npx skills add obra/superpowers"
 }
}
```

Теперь `npm run skills:install` устанавливает весь стек одной командой.

Способ 3 — ручная установка (fallback)

```
git clone --depth 1 https://github.com/vercel-labs/agent-skills.git /tmp/vercel-agent-skills
cp -r /tmp/vercel-agent-skills/skills/agent-skills-cli ~/.claude/skills/agent-skills-cli
```

Проверка установки

```
npx skills --version
npx skills list
```

Быстрый старт: как установить все 9 скиллов

Промпт:

~~~text
Prompt for Cloud Cowork / Cloud AI: Install Agent Skills into Cloud AI and Coding-Agent Directories

You are a senior DevOps + AI-agent tooling engineer working inside my Cloud Cowork / Cloud AI workspace.

Objective

Install the skills listed below in two places at the same time:

1. The active Cloud AI / Cloud Cowork skill directory used by the current assistant environment.
2. A separate shared directory for code agents such as Claude Code, Cursor, Codex, Gemini CLI, OpenCode, Anti-Gravity, and similar tools.

The installation must be safe, reproducible, verified, and documented.

Important assumptions

- Assume this is a Linux/macOS-like development environment with shell access.
- Do not overwrite existing skills without backing them up.
- Do not rely on one hardcoded path unless the environment clearly uses it.
- Prefer official installation methods first.
- If a skill has no direct CLI install command, fetch or copy its `SKILL.md` and required assets into the correct skill folder manually.
- If a repository, URL, package, or command is unavailable, report it clearly and continue installing the remaining skills.
- Before changing files, inspect the environment and identify likely skill directories.

Target installation locations

Detect and use the best available directories.

Check these possible locations and use whichever are valid in this environment:

Cloud AI / Cloud Cowork skill directory

Look for existing skill directories in places like:

- `~/.claude/skills`
- `~/.config/claude/skills`
- `~/.cloud-ai/skills`
- `~/.cloud-cowork/skills`
- `./.claude/skills`
- `./.skills`
- Any workspace-specific skill directory mentioned in local docs, config files, environment variables, or agent settings.

Shared code-agent skill directory

Create or use:

- `./agent-skills`
- `./.agent-skills`
- `~/.agent-skills`
- Or another clearly better existing shared agent-skill directory if the workspace already has one.

Use a project-local shared directory by default unless the environment clearly expects a global directory.

Skills to install

Block 1 — Frontend and design

1. `frontend-design`
  - Source: Anthropic official Claude Code plugin
  - GitHub: `https://github.com/anthropics/claude-code/tree/main/plugins/frontend-design`
  - Install command:
   ```bash
   npx skills add anthropics/claude-code skill frontend-design
   ```
  - Alternative Claude Code command:
   ```text
   /plugin install frontend-design@claude-plugins-official
   ```

2. `web-design-guidelines`
  - Source: Vercel Labs agent skills
  - GitHub: `https://github.com/vercel-labs/agent-skills/blob/main/skills/web-design-guidelines/SKILL.md`
  - Install command:
   ```bash
   npx skills add vercel-labs/agent-skills
   ```

3. `shadcn-ui`
  - Source: shadcn official
  - Docs: `https://ui.shadcn.com/docs/skills`
  - Project-aware skill; must read `components.json` if present.

Block 2 — Power tools for Claude Code

4. `superpowers`
  - Source: obra / Jesse Vincent
  - GitHub repo: `https://github.com/obra/superpowers`
  - Marketplace: `https://github.com/obra/superpowers-marketplace`
  - Skills: `https://github.com/obra/superpowers-skills`
  - Claude plugin page: `https://claude.com/plugins/superpowers`
  - Commands to verify after install:
   ```text
   /brainstorm
   /write-plan
   /execute-plan
   ```

5. `react-best-practices`
  - Source: Vercel Labs
  - GitHub: `https://github.com/vercel-labs/agent-skills/blob/main/skills/react-best-practices/SKILL.md`
  - Install command:
   ```bash
   npx skills add vercel-labs/agent-skills
   ```

6. `postgres-best-practices`
  - Source: Supabase
  - Listing: `https://mcpservers.org/agent-skills/supabase/postgres-best-practices`
  - Scope: performance, zero-downtime migrations, RLS.

Block 3 — Unique skills

7. `mcp-builder`
  - Source: Anthropic official
  - Repo: `https://github.com/anthropics/skills`
  - Purpose: builds MCP servers in Python or TypeScript.

8. `canvas-design`
  - Source: Anthropic / community listing
  - GitHub: `https://github.com/ComposioHQ/awesome-claude-skills/blob/master/canvas-design/SKILL.md`
  - Related repo: `https://github.com/anthropics/skills`
  - Purpose: generates PNG/PDF through code, not Figma.

9. `agent-skills CLI`
  - Source: Vercel Labs
  - GitHub: `https://github.com/vercel-labs/agent-skills`
  - CLI/package: `https://github.com/vercel-labs/skills`
  - Install pattern:
   ```bash
   npx skills add <org>/<repo>
   ```
  - Must support installation into directories used by Claude Code, Cursor, Codex, Gemini CLI, OpenCode, and 50+ agent tools when possible.

Required workflow

Step 1 — Inspect environment

Run safe read-only checks first:

```bash
pwd
ls -la
env | sort | grep -Ei 'claude|cloud|cowork|agent|skill|cursor|codex|gemini|openai|anthropic' || true
find . -maxdepth 4 -iname '*skill*' -o -iname 'SKILL.md' 2>/dev/null | head -100
find ~ -maxdepth 4 -type d $begin:math:text$ \-iname \'\*claude\*\' \-o \-iname \'\*skill\*\' \-o \-iname \'\*agent\*\' $end:math:text$ 2>/dev/null | head -100
```

Then identify:

- Current workspace root.
- Existing Cloud AI skill directory, if any.
- Existing Claude Code skill/plugin directory, if any.
- Existing shared agent skill directory, if any.
- Whether `node`, `npm`, `npx`, `git`, `python`, and `curl` are available.

Step 2 — Prepare directories

Create target directories only after inspection.

Use a structure similar to:

```
<cloud-ai-skills-dir>/
  frontend-design/
  web-design-guidelines/
  shadcn-ui/
  superpowers/
  react-best-practices/
  postgres-best-practices/
  mcp-builder/
  canvas-design/
  agent-skills-cli/

<shared-agent-skills-dir>/
  frontend-design/
  web-design-guidelines/
  shadcn-ui/
  superpowers/
  react-best-practices/
  postgres-best-practices/
  mcp-builder/
  canvas-design/
  agent-skills-cli/
```

If a target skill folder already exists:

1. Check whether it already contains a valid `SKILL.md`.
2. Back it up before replacing or modifying:

```bash
cp -a existing-folder existing-folder.backup.$(date +%Y%m%d-%H%M%S)
```

1. Prefer updating in place only if it is safe.

Step 3 — Install using official tools where possible

Use `npx skills` for repositories that support it.

Run commands defensively and capture output:

```bash
npx skills --help || true
npx skills add anthropics/claude-code skill frontend-design || true
npx skills add vercel-labs/agent-skills || true
```

If `npx skills add` installs into a default agent directory, locate the installed files and copy or sync the relevant skill folders into both target directories.

Do not assume the install succeeded unless `SKILL.md` files are present.

Step 4 — Manual fallback installation

For any skill not installed automatically:

1. Clone or fetch the official repository.
2. Locate the relevant skill folder or `SKILL.md`.
3. Copy the skill folder into both target directories.
4. Preserve all required companion files, examples, scripts, and assets.
5. Ensure each installed skill has a clear `SKILL.md`.

Example fallback pattern:

```bash
mkdir -p /tmp/agent-skill-install
cd /tmp/agent-skill-install

git clone --depth 1 https://github.com/vercel-labs/agent-skills.git vercel-agent-skills || true
git clone --depth 1 https://github.com/anthropics/skills.git anthropics-skills || true
git clone --depth 1 https://github.com/obra/superpowers.git superpowers || true
git clone --depth 1 https://github.com/obra/superpowers-skills.git superpowers-skills || true
```

Then search:

```bash
find /tmp/agent-skill-install -iname 'SKILL.md' -o -type d -iname '*frontend*' -o -type d -iname '*react*' -o -type d -iname '*postgres*' -o -type d -iname '*mcp*' -o -type d -iname '*canvas*' -o -type d -iname '*shadcn*'
```

Step 5 — Normalize installed skill folders

Each installed skill folder must contain at minimum:

```
skill-name/
  SKILL.md
```

Where useful, also include:

```
skill-name/
  README.md
  examples/
  scripts/
  assets/
  metadata.json
```

If `metadata.json` is missing, create a simple metadata file:

```json
{
  "name": "skill-name",
  "source": "source-url",
  "installed_for": ["cloud-ai", "coding-agents"],
  "status": "installed"
}
```

Do not invent false documentation. If a field is unknown, write `"unknown"`.

Step 6 — Create an installation manifest

Create a manifest file at:

```
<shared-agent-skills-dir>/INSTALL_MANIFEST.md
```

The manifest must include:

- Date and environment.
- Detected Cloud AI skill directory.
- Detected shared agent skill directory.
- Installed skill list.
- Source URL for each skill.
- Install method:
   - `npx skills`
   - cloned repo
   - direct file fetch
   - manual fallback
- Whether `SKILL.md` exists.
- Any failures or partial installs.
- Next actions required by the user.

Also create machine-readable JSON:

```
<shared-agent-skills-dir>/install-manifest.json
```

Step 7 — Validate installation

Run validation checks:

```bash
find "<cloud-ai-skills-dir>" -maxdepth 3 -name "SKILL.md" -print
find "<shared-agent-skills-dir>" -maxdepth 3 -name "SKILL.md" -print
```

For each expected skill, confirm:

- Folder exists.
- `SKILL.md` exists.
- File is non-empty.
- Source is documented.
- Skill is present in both target locations.

Create a final validation table:

Skill	**Cloud AI installed**	**Shared agent dir installed**	**SKILL.md found**	**Method**	**Notes**

Step 8 — Configure agent discovery if possible

If this environment supports agent configuration files, add or update configuration safely.

Look for files such as:

- `.claude/settings.json`
- `.claude/config.json`
- `.cursor/`
- `.codex/`
- `.gemini/`
- `.openhands/`
- `.opencode/`
- Any Cloud Cowork / Cloud AI config.

Do not break existing config.

If configuration format is unclear:

- Do not edit it blindly.
- Instead, create a proposed config snippet in:

```
<shared-agent-skills-dir>/AGENT_CONFIG_SNIPPETS.md
```

The snippet should show how to point agents to the shared skill directory.

Step 9 — Final report

Return a concise final report with:

1. What was installed successfully.
2. What was installed partially.
3. What failed and why.
4. Exact directories used.
5. Validation table.
6. Commands the user can run to verify.
7. Any manual activation steps needed inside Claude Code or another agent.

Output requirements

Do not just describe what should be done. Actually execute the safe installation steps available in this environment.

Use shell commands where appropriate.

Be careful with destructive commands:

- Do not use `rm -rf` on user directories.
- Do not overwrite existing files without backup.
- Do not modify global configs unless the correct format is obvious.
- Do not store secrets.
- Do not expose environment variables containing tokens.

Completion criteria

The task is complete only when:

- All possible skills are installed or clearly reported as failed.
- Both Cloud AI and shared agent directories are populated.
- Every installed skill has a `SKILL.md`.
- A human-readable manifest exists.
- A JSON manifest exists.
- A final validation table is provided.

Важно: не экономь токены. Если не влазит в одно сообщение — раздели на несколько.

Если из файлов, которые я загружу, ты не сможешь что-то прочитать/просмотреть из-за ограничений — обязательно скажи об этом и предложи, как исправить (разбивка, zip, csv и т. п.).
~~~