Claude ✳

الدليل الكامل لبناء المهارات (Skills) لكلود

نسخة عربية مترجمة بتصرف من كتيب Anthropic الرسمي «The Complete Guide to Building Skills for Claude».
المصطلحات التقنية مُترجَمة مع إبقاء الأصل الإنجليزي بين قوسين عند أول ورود؛ وأمثلة الشيفرة (code) تُركت بالإنجليزية كما هي.

الملف الأصلي موجود على الرابط

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

استخدم معالج صناعة المهارات الألي على منصة أوامر مجانا عبر هذا الرابط

https://awamer.ai/ar/ai-agent-skills/wizard

الترجمة بواسطة منصة أوامر

Awamer.ai

المحتويات

المقدمة

المهارة (Skill) هي مجموعة من التعليمات — مُحزَّمة في صورة مجلد بسيط — تُعلّم كلود (Claude) كيفية التعامل مع مهام أو سير عمل (workflows) محدّدة. تُعدّ المهارات من أقوى الطرق لتخصيص كلود حسب احتياجاتك الخاصة. فبدلاً من إعادة شرح تفضيلاتك وإجراءاتك وخبرتك في مجالك في كل محادثة، تتيح لك المهارات أن تُعلّم كلود مرة واحدة وتستفيد منها في كل مرة.

تتجلّى قوة المهارات عندما يكون لديك سير عمل متكرر: توليد تصميمات واجهات أمامية (frontend) من مواصفات، أو إجراء بحث وفق منهجية ثابتة، أو إنشاء مستندات تتبع دليل الأسلوب (style guide) لفريقك، أو تنسيق عمليات متعددة الخطوات. وهي تعمل جيداً مع القدرات المدمجة في كلود مثل تنفيذ الشيفرة وإنشاء المستندات. ولمن يبنون تكاملات MCP، تضيف المهارات طبقة قوية إضافية تساعد على تحويل الوصول الخام للأدوات إلى سير عمل موثوق ومُحسَّن.

يغطّي هذا الدليل كل ما تحتاج معرفته لبناء مهارات فعّالة — من التخطيط والبنية إلى الاختبار والتوزيع. وسواء كنت تبني مهارة لنفسك أو لفريقك أو للمجتمع، ستجد أنماطاً عملية وأمثلة من الواقع في كل صفحاته.

ما الذي ستتعلّمه:

- المتطلبات التقنية وأفضل الممارسات لبنية المهارة.
- أنماط المهارات المستقلة وسير العمل المعزّز بـ MCP.
- الأنماط التي رأينا أنها تعمل جيداً عبر حالات استخدام مختلفة.
- كيفية اختبار مهاراتك وتحسينها وتوزيعها.

لمن هذا الدليل:

- المطوّرون الذين يريدون من كلود اتباع سير عمل محدّد بثبات.
- المستخدمون المتمكّنون (power users) الذين يريدون من كلود اتباع سير عمل معيّن.
- الفِرَق التي تسعى إلى توحيد طريقة عمل كلود عبر مؤسستها.

مساران عبر هذا الدليل

هل تبني مهارات مستقلة؟ ركّز على «الأساسيات» و«التخطيط والتصميم» والفئتين 1 و2. أمّا إن كنت تعزّز تكامل MCP، فقسم «المهارات + MCP» والفئة 3 هما لك. كلا المسارين يتشاركان المتطلبات التقنية نفسها، لكنك تختار ما يناسب حالة استخدامك.

ما الذي ستجنيه من هذا الدليل: في النهاية ستكون قادراً على بناء مهارة وظيفية في جلسة واحدة. توقّع نحو 15 إلى 30 دقيقة لبناء واختبار أول مهارة عاملة باستخدام منشئ المهارات (skill-creator).

هيّا نبدأ.

الفصل 1

الأساسيات

ما المهارة؟

المهارة هي مجلد يحتوي على:

- SKILL.md (مطلوب): تعليمات بصيغة Markdown مع ترويسة YAML (YAML frontmatter).
- scripts/ (اختياري): شيفرة قابلة للتنفيذ (Python، Bash، إلخ).
- references/ (اختياري): توثيق يُحمَّل عند الحاجة.
- assets/ (اختياري): قوالب وخطوط وأيقونات تُستخدَم في المُخرَجات.

مبادئ التصميم الأساسية

الكشف التدريجي (Progressive Disclosure)

تستخدم المهارات نظاماً من ثلاثة مستويات:

- المستوى الأول (ترويسة YAML): يُحمَّل دائماً ضمن موجّه النظام (system prompt) لكلود. يوفّر معلومات كافية فقط ليعرف كلود متى ينبغي استخدام كل مهارة دون تحميلها كاملة في السياق.
- المستوى الثاني (متن SKILL.md): يُحمَّل عندما يرى كلود أن المهارة ملائمة للمهمة الحالية. يحتوي على التعليمات والإرشادات الكاملة.
- المستوى الثالث (الملفات المرتبطة): ملفات إضافية مُحزَّمة داخل مجلد المهارة يستطيع كلود اختيار تصفّحها واستكشافها عند الحاجة فقط.

يقلّل هذا الكشف التدريجي من استهلاك الرموز (tokens) مع الحفاظ على الخبرة المتخصّصة.

قابلية التركيب (Composability)

يستطيع كلود تحميل عدّة مهارات في آنٍ واحد. ينبغي أن تعمل مهارتك جيداً جنباً إلى جنب مع غيرها، لا أن تفترض أنها القدرة الوحيدة المتاحة.

قابلية النقل (Portability)

تعمل المهارات بالطريقة نفسها عبر Claude.ai و Claude Code وواجهة البرمجة (API). أنشئ المهارة مرة واحدة وتعمل عبر جميع الواجهات دون تعديل، شريطة أن تدعم البيئة أيّ اعتماديات (dependencies) تتطلبها المهارة.

لِبُنَاة MCP: المهارات + الموصِّلات (Connectors)

💡 هل تبني مهارات مستقلة دون MCP؟ انتقل إلى «التخطيط والتصميم» — يمكنك دائماً العودة هنا لاحقاً.

إن كان لديك خادم MCP عامل، فقد أنجزت الجزء الصعب. المهارات هي طبقة المعرفة فوقه — تلتقط سير العمل وأفضل الممارسات التي تعرفها مسبقاً، حتى يطبّقها كلود بثبات.

تشبيه المطبخ

يوفّر MCP المطبخ الاحترافي: الوصول إلى الأدوات والمكوّنات والمعدّات.

توفّر المهارات الوصفات: تعليمات خطوة بخطوة لإعداد شيء ذي قيمة.

معاً يمكّنان المستخدمين من إنجاز مهام معقّدة دون الحاجة إلى استنباط كل خطوة بأنفسهم.

كيف يعملان معاً:

MCP (الاتصال)	المهارات (المعرفة)
يصِل كلود بخدمتك (Linear، Asana، Notion، إلخ).	يُعلّم كلود كيفية استخدام خدمتك بفعالية.
يوفّر وصولاً للبيانات بالزمن الحقيقي واستدعاء الأدوات.	يلتقط سير العمل وأفضل الممارسات.
ما الذي يستطيع كلود فعله.	كيف ينبغي لكلود أن يفعله.

لماذا يهمّ هذا لمستخدمي MCP لديك

من دون مهارات:

- يوصِل المستخدمون MCP الخاص بك لكنهم لا يعرفون ما الخطوة التالية.
- تذاكر دعم تسأل «كيف أفعل كذا باستخدام تكاملك؟».
- كل محادثة تبدأ من الصفر.
- نتائج غير متّسقة لأن المستخدمين يوجّهون بطرق مختلفة في كل مرة.
- يلوم المستخدمون موصِّلك بينما المشكلة الحقيقية هي غياب إرشاد سير العمل.

مع المهارات:

- سير عمل جاهز يُفعَّل تلقائياً عند الحاجة.
- استخدام موثوق ومتّسق للأدوات.
- أفضل الممارسات مغروسة في كل تفاعل.
- منحنى تعلّم أقل لتكاملك.

التخطيط والتصميم

ابدأ بحالات الاستخدام (Use Cases)

قبل كتابة أي شيفرة، حدّد 2 إلى 3 حالات استخدام ملموسة يُفترض أن تتيحها مهارتك.

تعريف جيّد لحالة استخدام:

```
Use Case: Project Sprint Planning
Trigger: User says "help me plan this sprint" or "create
sprint tasks"
Steps:
1. Fetch current project status from Linear (via MCP)
2. Analyze team velocity and capacity
3. Suggest task prioritization
4. Create tasks in Linear with proper labels and estimates
Result: Fully planned sprint with tasks created
```

اسأل نفسك:

- ما الذي يريد المستخدم إنجازه؟
- ما سير العمل متعدد الخطوات الذي يتطلبه ذلك؟
- ما الأدوات المطلوبة (مدمجة أم عبر MCP)؟
- ما معرفة المجال أو أفضل الممارسات التي ينبغي غرسها؟

فئات شائعة لحالات استخدام المهارات

في Anthropic رصدنا ثلاث فئات شائعة لحالات الاستخدام:

الفئة 1: إنشاء المستندات والأصول (Document & Asset Creation)

تُستخدَم لِـ: إنشاء مُخرَجات عالية الجودة ومتّسقة، تشمل المستندات والعروض التقديمية والتطبيقات والتصميمات والشيفرة، إلخ.

مثال واقعي: مهارة frontend-design (انظر أيضاً مهارات docx و pptx و xlsx و ppt).

«أنشئ واجهات أمامية مميّزة بجودة إنتاجية عالية وبمعايير تصميم رفيعة. استخدمها عند بناء مكوّنات الويب والصفحات والأصول والملصقات أو التطبيقات.»

تقنيات أساسية:

- غرس أدلّة الأسلوب ومعايير العلامة التجارية.
- بنى قوالب للحصول على مُخرَج متّسق.
- قوائم تحقق للجودة قبل الإنهاء.
- لا حاجة لأدوات خارجية — يستخدم قدرات كلود المدمجة.

الفئة 2: أتمتة سير العمل (Workflow Automation)

تُستخدَم لِـ: العمليات متعددة الخطوات التي تستفيد من منهجية ثابتة، بما في ذلك التنسيق عبر خوادم MCP متعددة.

مثال واقعي: مهارة skill-creator.

«دليل تفاعلي لإنشاء مهارات جديدة. يرشد المستخدم عبر تعريف حالة الاستخدام، وتوليد الترويسة، وكتابة التعليمات، والتحقق.»

تقنيات أساسية:

- سير عمل خطوة بخطوة مع بوّابات تحقّق (validation gates).
- قوالب للبُنى الشائعة.
- مراجعة واقتراحات تحسين مدمجة.
- حلقات تحسين تكرارية.

الفئة 3: تعزيز MCP (MCP Enhancement)

تُستخدَم لِـ: إرشاد سير العمل لتعزيز الوصول للأدوات الذي يوفّره خادم MCP.

مثال واقعي: مهارة sentry-code-review (من Sentry).

«تحلّل تلقائياً وتُصلِح العلل المكتشفة في طلبات الدمج (Pull Requests) على GitHub باستخدام بيانات مراقبة الأخطاء من Sentry عبر خادم MCP الخاص بهم.»

تقنيات أساسية:

- تنسيق استدعاءات MCP المتعددة بالتسلسل.
- غرس خبرة المجال.
- توفير سياق وإلّا فسيحتاج المستخدم لتحديده.
- معالجة الأخطاء لمشكلات MCP الشائعة.

حدّد معايير النجاح

كيف ستعرف أن مهارتك تعمل؟

هذه أهداف طموحة — معايير تقريبية لا عتبات صارمة. اسعَ إلى الدقّة لكن تقبّل وجود عنصر من التقييم القائم على «الإحساس». نحن نطوّر فعلياً إرشادات وأدوات قياس أكثر متانة.

مقاييس كمّية:

- تُفعَّل المهارة في 90% من الاستعلامات ذات الصلة.
  - كيف تقيس: شغّل 10 إلى 20 استعلاماً اختبارياً يُفترض أن تُفعِّل مهارتك. تتبّع كم مرة تُحمَّل تلقائياً مقابل ما يتطلب استدعاءً صريحاً.
- إتمام سير العمل في عدد X من استدعاءات الأدوات.
  - كيف تقيس: قارن المهمة نفسها مع المهارة وبدونها. عُدّ استدعاءات الأدوات وإجمالي الرموز المستهلكة.
- صفر استدعاءات API فاشلة لكل سير عمل.
  - كيف تقيس: راقب سجلات خادم MCP أثناء التشغيل الاختباري. تتبّع معدلات إعادة المحاولة ورموز الأخطاء.

مقاييس نوعية:

- لا يحتاج المستخدمون إلى توجيه كلود بشأن الخطوات التالية.
  - كيف تُقيّم: أثناء الاختبار، لاحظ كم مرة تحتاج إلى إعادة التوجيه أو التوضيح. اطلب التغذية الراجعة من مستخدمين تجريبيين.
- إتمام سير العمل دون تصحيح من المستخدم.
  - كيف تُقيّم: شغّل الطلب نفسه 3 إلى 5 مرات. قارن المُخرَجات من حيث الاتساق البنيوي والجودة.
- نتائج متّسقة عبر الجلسات.
  - كيف تُقيّم: هل يستطيع مستخدم جديد إنجاز المهمة من المحاولة الأولى بأقل إرشاد؟

المتطلبات التقنية

بنية الملفات

```
your-skill-name/
├── SKILL.md              # Required - main skill file
├── scripts/              # Optional - executable code
│   ├── process_data.py   # Example
│   └── validate.sh       # Example
├── references/           # Optional - documentation
│   ├── api-guide.md      # Example
│   └── examples/         # Example
└── assets/               # Optional - templates, etc.
    └── report-template.md # Example
```

قواعد حاسمة

تسمية SKILL.md:

- يجب أن يكون الاسم بالضبط SKILL.md (حسّاس لحالة الأحرف).
- لا يُقبل أي تنويع (مثل SKILL.MD أو skill.md).

تسمية مجلد المهارة:

- استخدم صيغة الشَّرطة notion-project-setup :(kebab-case) ✅
- بلا مسافات: Notion Project Setup ❌
- بلا شُرَط سفلية: notion_project_setup ❌
- بلا أحرف كبيرة: NotionProjectSetup ❌

لا تضع README.md:

- لا تُضمِّن README.md داخل مجلد المهارة.
- كل التوثيق يذهب إلى SKILL.md أو references/.
- ملاحظة: عند التوزيع عبر GitHub ستظل تريد ملف README على مستوى المستودع للبشر — انظر «التوزيع والمشاركة».

ترويسة YAML: الجزء الأهم

ترويسة YAML هي الطريقة التي يقرّر بها كلود إن كان سيُحمّل مهارتك. أتقِنها.

الصيغة الدنيا المطلوبة:

```
---
name: your-skill-name
description: What it does. Use when user asks to [specific
phrases].
---
```

هذا كل ما تحتاجه للبدء.

متطلبات الحقول

name (مطلوب):

- صيغة الشَّرطة (kebab-case) فقط.
- بلا مسافات أو أحرف كبيرة.
- ينبغي أن يطابق اسم مجلد المهارة.

description (مطلوب):

- يجب أن يتضمّن كليهما: ما الذي تفعله المهارة + متى تُستخدَم (شروط التفعيل/trigger conditions).
- أقل من 1024 حرفاً.
- بلا وسوم XML.
- أدرِج المهام المحدّدة التي قد يذكرها المستخدمون.
- اذكر أنواع الملفات إن كانت ذات صلة.

license (اختياري): استخدمه إن كنت تجعل المهارة مفتوحة المصدر. شائع: MIT، Apache-2.0.

compatibility (اختياري): من 1 إلى 500 حرف. يشير إلى متطلبات البيئة، مثل المنتج المستهدف، حِزم النظام المطلوبة، احتياجات الوصول للشبكة، إلخ.

metadata (اختياري): أي أزواج مفتاح-قيمة مخصّصة. المقترَح: المؤلّف، الإصدار، خادم MCP.

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

قيود الأمان

ممنوع في الترويسة:

- أقواس XML الزاوية — قيد أمني.
- المهارات التي تحمل «claude» أو «anthropic» في الاسم (محجوزة).

لماذا: تظهر الترويسة في موجّه نظام كلود؛ قد يحقن المحتوى الخبيث تعليمات.

كتابة مهارات فعّالة

حقل الوصف (description)

بحسب مدوّنة الهندسة في Anthropic: «توفّر هذه البيانات الوصفية… معلومات كافية فقط ليعرف كلود متى ينبغي استخدام كل مهارة دون تحميلها كاملة في السياق.» هذا هو المستوى الأول من الكشف التدريجي.

البنية:

```
[ما الذي تفعله] + [متى تُستخدَم] + [القدرات الأساسية]
```

أمثلة على أوصاف جيّدة:

```
# Good - specific and actionable
description: Analyzes Figma design files and generates
developer handoff documentation. Use when user uploads .fig
files, asks for "design specs", "component documentation", or
"design-to-code handoff".

# Good - includes trigger phrases
description: Manages Linear project workflows including sprint
planning, task creation, and status tracking. Use when user
mentions "sprint", "Linear tasks", "project planning", or asks
to "create tickets".

# Good - clear value proposition
description: End-to-end customer onboarding workflow for
PayFlow. Handles account creation, payment setup, and
subscription management. Use when user says "onboard new
customer", "set up subscription", or "create PayFlow account".
```

أمثلة على أوصاف سيّئة:

```
# Too vague
description: Helps with projects.

# Missing triggers
description: Creates sophisticated multi-page documentation
systems.

# Too technical, no user triggers
description: Implements the Project entity model with
hierarchical relationships.
```

كتابة التعليمات الرئيسية

بعد الترويسة، اكتب التعليمات الفعلية بصيغة Markdown.

البنية المقترَحة: كيّف هذا القالب لمهارتك، واستبدِل الأقسام بين الأقواس بمحتواك المحدّد.

```
---
name: your-skill
```

```
description: [...]
---

# Your Skill Name

## Instructions

### Step 1: [First Major Step]
Clear explanation of what happens.

Example:
```bash
python scripts/fetch_data.py --project-id PROJECT_ID
Expected output: [describe what success looks like]
```
(Add more steps as needed)

Examples

Example 1: [common scenario]
User says: "Set up a new marketing campaign"
Actions:
1. Fetch existing campaigns via MCP
2. Create new campaign with provided parameters
Result: Campaign created with provided link

Troubleshooting

Error: [Common error message]
Cause: [Why it happens]
Solution: [How to fix]
```

أفضل الممارسات للتعليمات

كن محدّداً وقابلاً للتنفيذ

✅ **جيّد:**

```
Run `python scripts/validate.py --input {filename}` to check
data format.
If validation fails, common issues include:
- Missing required fields (add them to the CSV)
- Invalid date formats (use YYYY-MM-DD)
```

❌ **سيّئ:**

```
Validate the data before proceeding.
```

أدرِج معالجة الأخطاء

```
## Common Issues

### MCP Connection Failed:
If you see "Connection refused":
```

```
1. Verify MCP server is running: Check Settings > Extensions
2. Confirm API key is valid
3. Try reconnecting: Settings > Extensions > [Service] >
Reconnect
```

أشِر إلى الموارد المُحزَّمة بوضوح

```
Before writing queries, consult `references/api-patterns.md`
for:
- Rate limiting guidance
- Pagination patterns
- Error codes and handling
```

استخدم الكشف التدريجي

أبقِ SKILL.md مركّزاً على التعليمات الجوهرية. انقل التوثيق التفصيلي إلى references/ واربط إليه. (انظر «مبادئ التصميم الأساسية» لكيفية عمل نظام المستويات الثلاثة.)

الاختبار والتحسين التكراري

يمكن اختبار المهارات بمستويات متفاوتة من الصرامة حسب احتياجاتك:

- الاختبار اليدوي في Claude.ai: شغّل الاستعلامات مباشرة ولاحظ السلوك. تكرار سريع، لا حاجة لإعداد.
- الاختبار المُؤتمَت في Claude Code: أتمِت حالات الاختبار لتحقّق متكرر عبر التغييرات.
- الاختبار البرمجي عبر واجهة المهارات (skills API): ابنِ مجموعات تقييم تعمل بانتظام مقابل مجموعات محدّدة.

اختر الأسلوب الذي يطابق متطلبات الجودة لديك ومدى ظهور مهارتك. فالمهارة المستخدمة داخلياً لفريق صغير لها احتياجات اختبار مختلفة عن مهارة منشورة لآلاف المستخدمين في مؤسسة.

نصيحة احترافية: كرّر على مهمة واحدة قبل التوسّع. وجدنا أن أكثر منشئي المهارات فعالية يكرّرون على مهمة واحدة صعبة حتى ينجح فيها كلود، ثم يستخلصون الأسلوب الناجح في مهارة. يستفيد هذا من التعلّم داخل السياق (in-context learning) لدى كلود ويوفّر إشارة أسرع من الاختبار الواسع. وبمجرد أن تكون لديك قاعدة عاملة، توسّع إلى حالات اختبار متعددة للتغطية.

أسلوب الاختبار المُوصى به

بناءً على التجربة المبكّرة، يغطّي اختبار المهارات الفعّال عادةً ثلاثة مجالات:

1. اختبارات التفعيل (Triggering)

الهدف: ضمان تحميل مهارتك في الأوقات الصحيحة.

حالات الاختبار:

- ✅ تُفعَّل عند المهام الواضحة.
- ✅ تُفعَّل عند الطلبات المُعاد صياغتها.
- ❌ لا تُفعَّل عند الموضوعات غير ذات الصلة.

مثال على مجموعة اختبار:

```
Should trigger:
- "Help me set up a new ProjectHub workspace"
- "I need to create a project in ProjectHub"
- "Initialize a ProjectHub project for Q4 planning"

Should NOT trigger:
- "What's the weather in San Francisco?"
- "Help me write Python code"
- "Create a spreadsheet" (unless ProjectHub handles sheets)
```

2. الاختبارات الوظيفية (Functional)

الهدف: التحقّق من أن المهارة تنتج مُخرَجات صحيحة.

حالات الاختبار:

- توليد مُخرَجات صحيحة.
- نجاح استدعاءات API.
- عمل معالجة الأخطاء.
- تغطية الحالات الحدّية (edge cases).

```
Test: Create project with 5 tasks
Given: Project name "Q4 Planning", 5 task descriptions
When: Skill executes workflow
Then:
   - Project created in ProjectHub
   - 5 tasks created with correct properties
   - All tasks linked to project
   - No API errors
```

3. مقارنة الأداء (Performance comparison)

الهدف: إثبات أن المهارة تحسّن النتائج مقارنةً بخط الأساس (baseline).

استخدم المقاييس من «حدّد معايير النجاح». إليك ما قد تبدو عليه المقارنة:

```
Without skill:
- User provides instructions each time
- 15 back-and-forth messages
- 3 failed API calls requiring retry
- 12,000 tokens consumed

With skill:
- Automatic workflow execution
- 2 clarifying questions only
- 0 failed API calls
- 6,000 tokens consumed
```

استخدام مهارة منشئ المهارات (skill-creator)

مهارة skill-creator — المتاحة في Claude.ai عبر دليل الإضافات (plugin directory) أو للتنزيل في Claude Code — تساعدك على بناء المهارات والتكرار عليها. إن كان لديك خادم MCP وتعرف أهم 2 إلى 3 من سير عملك، يمكنك بناء واختبار مهارة وظيفية في جلسة واحدة — غالباً خلال 15 إلى 30 دقيقة.

إنشاء المهارات:

- توليد مهارات من أوصاف باللغة الطبيعية.
- إنتاج SKILL.md منسّق بشكل صحيح مع الترويسة.
- اقتراح عبارات التفعيل والبنية.

مراجعة المهارات:

- الإشارة إلى المشكلات الشائعة (أوصاف غامضة، تفعيلات مفقودة، مشكلات بنيوية).
- تحديد مخاطر التفعيل الزائد/الناقص المحتملة.
- اقتراح حالات اختبار بناءً على الغرض المعلَن للمهارة.

التحسين التكراري:

- بعد استخدام مهارتك ومصادفة حالات حدّية أو إخفاقات، أعِد تلك الأمثلة إلى منشئ المهارات.
- مثال: «استخدم المشكلة والحل المُحدَّدين في هذه المحادثة لتحسين كيفية تعامل المهارة مع [الحالة الحدّية المحدّدة]».

للاستخدام:

```
"Use the skill-creator skill to help me build a skill for
[your use case]"
```

ملاحظة: يساعدك منشئ المهارات على تصميم المهارات وتحسينها، لكنه لا يشغّل مجموعات اختبار مُؤتمَتة ولا ينتج نتائج تقييم كمّية.

التكرار بناءً على التغذية الراجعة

المهارات مستندات حيّة. خطّط للتكرار بناءً على:

إشارات التفعيل الناقص (Undertriggering):

- لا تُحمَّل المهارة حين ينبغي ذلك.
- يفعّلها المستخدمون يدوياً.
- أسئلة دعم حول متى تُستخدَم.

الحل: أضِف مزيداً من التفصيل والدقّة إلى الوصف — قد يشمل ذلك كلمات مفتاحية خاصةً للمصطلحات التقنية.

إشارات التفعيل الزائد (Overtriggering):

- تُحمَّل المهارة لاستعلامات غير ذات صلة.
- يعطّلها المستخدمون.
- التباس حول الغرض.

الحل: أضِف تفعيلات سلبية (negative triggers)، وكن أكثر تحديداً.

مشكلات التنفيذ:

- نتائج غير متّسقة.
- إخفاق استدعاءات API.
- الحاجة إلى تصحيحات من المستخدم.

الحل: حسّن التعليمات، وأضِف معالجة للأخطاء.

التوزيع والمشاركة

تجعل المهارات تكامل MCP لديك أكثر اكتمالاً. وحين يقارن المستخدمون الموصِّلات، فإن تلك المزوّدة بمهارات تقدّم مساراً أسرع للقيمة، مانحةً إياك أفضلية على البدائل المعتمدة على MCP وحده.

نموذج التوزيع الحالي (يناير 2026)

كيف يحصل الأفراد على المهارات:

11. تنزيل مجلد المهارة.
12. ضغط المجلد (إن لزم).
13. الرفع إلى Claude.ai عبر: Settings > Capabilities > Skills.
14. أو وضعها في مجلد مهارات Claude Code.

على مستوى المؤسسة:

- يستطيع المسؤولون نشر المهارات على نطاق مساحة العمل (شُحِنت في 18 ديسمبر 2025).
- تحديثات تلقائية.
- إدارة مركزية.

معيار مفتوح

نشرنا Agent Skills كمعيار مفتوح. كما في MCP، نؤمن بأن المهارات ينبغي أن تكون قابلة للنقل عبر الأدوات والمنصّات — فالمهارة نفسها يجب أن تعمل سواء كنت تستخدم كلود أو منصّات ذكاء اصطناعي أخرى. ومع ذلك، صُمّمت بعض المهارات للاستفادة الكاملة من قدرات منصّة محدّدة؛ ويمكن للمؤلّفين الإشارة إلى ذلك في حقل compatibility الخاص بالمهارة. ونتعاون مع أعضاء النظام البيئي حول المعيار، ونتطلّع بحماس إلى التبنّي المبكّر.

استخدام المهارات عبر واجهة البرمجة (API)

لحالات الاستخدام البرمجية — مثل بناء التطبيقات أو الوكلاء (agents) أو سير العمل المُؤتمَت الذي يستفيد من واجهة البرمجة — توفّر واجهة البرمجة تحكّماً مباشراً في إدارة المهارات وتنفيذها.

قدرات أساسية:

- نقطة النهاية /v1/skills لعرض المهارات وإدارتها.
- إضافة المهارات إلى طلبات Messages API عبر مُعامِل container.skills.
- التحكّم بالإصدارات والإدارة عبر Claude Console.
- تعمل مع Claude Agent SDK لبناء وكلاء مخصّصين.

متى تستخدم المهارات عبر API مقابل Claude.ai:

حالة الاستخدام	الواجهة الأنسب
مستخدمون نهائيون يتفاعلون مع المهارات مباشرةً	Claude.ai / Claude Code
الاختبار اليدوي والتكرار أثناء التطوير	Claude.ai / Claude Code
سير عمل فردي أو وقتي (ad-hoc)	Claude.ai / Claude Code
تطبيقات تستخدم المهارات برمجياً	API
عمليات نشر إنتاجية على نطاق واسع	API
خطوط معالجة (pipelines) وأنظمة وكلاء مُؤتمَتة	API

ملاحظة: تتطلب المهارات في واجهة البرمجة أداة تنفيذ الشيفرة (Code Execution Tool) — وهي في مرحلة بيتا — والتي توفّر البيئة الآمنة التي تحتاجها المهارات للتشغيل.

للتفاصيل التنفيذية، انظر:

- دليل البدء السريع لواجهة المهارات (Skills API Quickstart).
- إنشاء مهارات مخصّصة (Create Custom skills).
- المهارات في Agent SDK.

الأسلوب المُوصى به اليوم

ابدأ باستضافة مهارتك على GitHub بمستودع عام، وملف README واضح (للزوار من البشر، والذي ينبغي ألّا يكون داخل مجلد المهارة)، ومثال استخدام مع لقطات شاشة. ثم أضِف قسماً إلى توثيق MCP الخاص بك يربط بالمهارة، ويشرح لماذا يكون استخدامهما معاً قيّماً، ويوفّر دليل بدء سريع.

1. الاستضافة على GitHub

- مستودع عام للمهارات مفتوحة المصدر.
- تعليمات تثبيت واضحة في README.
- مثال استخدام ولقطات شاشة.

2. التوثيق في مستودع MCP الخاص بك

- اربط بالمهارات من توثيق MCP.
- اشرح قيمة استخدامهما معاً.
- وفّر دليل بدء سريع.

3. أنشئ دليل تثبيت

```
## 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]"
```

تموضع مهارتك (Positioning)

تحدّد طريقة وصفك لمهارتك ما إذا كان المستخدمون سيفهمون قيمتها. سواء في README أو التوثيق أو التسويق، اجعل هذه المبادئ في ذهنك.

ركّز على النتائج لا الميزات:

✅ **جيّد:**

```
"The ProjectHub skill enables teams to set up complete project
workspaces in seconds — including pages, databases, and
templates — instead of spending 30 minutes on manual setup."
```

❌ **سيّئ:**

```
"The ProjectHub skill is a folder containing YAML frontmatter
and Markdown instructions that calls our MCP server tools."
```

أبرِز قصة MCP + المهارات:

```
"Our MCP server gives Claude access to your Linear projects.
Our skills teach Claude your team's sprint planning workflow.
Together, they enable AI-powered project management."
```

الأنماط ومعالجة المشكلات

برزت هذه الأنماط من مهارات أنشأها متبنّون مبكّرون وفِرَق داخلية. وهي تمثّل أساليب شائعة رأينا أنها تعمل جيداً، لا قوالب إلزامية.

اختيار أسلوبك: من المشكلة أم من الأداة؟

فكّر في الأمر كمتجر أدوات منزلية. قد تدخل بمشكلة — «أحتاج إصلاح خزانة المطبخ» — فيوجّهك موظّف إلى الأدوات الصحيحة. أو قد تختار مثقاباً جديداً وتسأل كيف تستخدمه لمهمتك المحدّدة.

تعمل المهارات بالطريقة نفسها:

- الأسلوب المنطلق من المشكلة (Problem-first): «أحتاج إعداد مساحة عمل لمشروع» ← مهارتك تنسّق استدعاءات MCP الصحيحة بالتسلسل الصحيح. يصف المستخدمون النتائج المرجوّة، والمهارة تتولّى الأدوات.
- الأسلوب المنطلق من الأداة (Tool-first): «لديّ Notion MCP موصول» ← مهارتك تُعلّم كلود سير العمل الأمثل وأفضل الممارسات. لدى المستخدمين وصول، والمهارة توفّر الخبرة.

معظم المهارات تميل إلى اتجاه واحد. معرفة أيّ إطار يناسب حالة استخدامك يساعدك على اختيار النمط الصحيح أدناه.

النمط 1: تنسيق سير عمل متسلسل

استخدمه عندما: يحتاج مستخدموك عمليات متعددة الخطوات بترتيب محدّد.

```
## Workflow: Onboard New Customer

### Step 1: Create Account
Call MCP tool: create_customer
Parameters: name, email, company

### Step 2: Setup Payment
Call MCP tool: setup_payment_method
Wait for: payment method verification

### Step 3: Create Subscription
Call MCP tool: create_subscription
Parameters: plan_id, customer_id (from Step 1)

### Step 4: Send Welcome Email
Call MCP tool: send_email
Template: welcome_email_template
```

تقنيات أساسية: ترتيب صريح للخطوات؛ اعتماديات بين الخطوات؛ تحقّق في كل مرحلة؛ تعليمات تراجع (rollback) عند الإخفاق.

النمط 2: التنسيق عبر MCP متعددة

استخدمه عندما: يمتدّ سير العمل عبر خدمات متعددة.

مثال: تسليم من التصميم إلى التطوير.

```
### Phase 1: Design Export (Figma MCP)
1. Export design assets from Figma
2. Generate design specifications
3. Create asset manifest

### Phase 2: Asset Storage (Drive MCP)
1. Create project folder in Drive
2. Upload all assets
3. Generate shareable links

### Phase 3: Task Creation (Linear MCP)
1. Create development tasks
2. Attach asset links to tasks
3. Assign to engineering team

### Phase 4: Notification (Slack MCP)
1. Post handoff summary to #engineering
2. Include asset links and task references
```

تقنيات أساسية: فصل واضح للمراحل؛ تمرير البيانات بين MCP؛ تحقّق قبل الانتقال للمرحلة التالية؛ معالجة أخطاء مركزية.

النمط 3: التحسين التكراري

استخدمه عندما: تتحسّن جودة المُخرَج مع التكرار.

مثال: توليد تقرير.

```
## Iterative Report Creation

### Initial Draft
1. Fetch data via MCP
2. Generate first draft report
3. Save to temporary file

### Quality Check
1. Run validation script: scripts/check_report.py
2. Identify issues:
    - Missing sections
    - Inconsistent formatting
    - Data validation errors

### Refinement Loop
1. Address each identified issue
2. Regenerate affected sections
3. Re-validate
4. Repeat until quality threshold met

### Finalization
1. Apply final formatting
2. Generate summary
3. Save final version
```

تقنيات أساسية: معايير جودة صريحة؛ تحسين تكراري؛ سكربتات تحقّق؛ معرفة متى تتوقف عن التكرار.

النمط 4: اختيار الأدوات الواعي بالسياق

استخدمه عندما: النتيجة نفسها لكن بأدوات مختلفة حسب السياق.

مثال: تخزين الملفات.

```
## Smart File Storage

### Decision Tree
1. Check file type and size
2. Determine best storage location:
    - Large files (>10MB): Use cloud storage MCP
    - Collaborative docs: Use Notion/Docs MCP
    - Code files: Use GitHub MCP
    - Temporary files: Use local storage

### Execute Storage
Based on decision:
- Call appropriate MCP tool
- Apply service-specific metadata
- Generate access link

### Provide Context to User
Explain why that storage was chosen
```

تقنيات أساسية: معايير قرار واضحة؛ خيارات بديلة (fallback)؛ شفافية بشأن الخيارات.

النمط 5: ذكاء خاص بالمجال

استخدمه عندما: تضيف مهارتك معرفة متخصّصة تتجاوز الوصول للأدوات.

مثال: الامتثال المالي.

```
## Payment Processing with Compliance

### Before Processing (Compliance Check)
1. Fetch transaction details via MCP
2. Apply compliance rules:
    - Check sanctions lists
    - Verify jurisdiction allowances
    - Assess risk level
3. Document compliance decision

### Processing
IF compliance passed:
   - Call payment processing MCP tool
   - Apply appropriate fraud checks
   - Process transaction
ELSE:
   - Flag for review
   - Create compliance case

### Audit Trail
```

```
- Log all compliance checks
- Record processing decisions
- Generate audit report
```

تقنيات أساسية: خبرة المجال مغروسة في المنطق؛ الامتثال قبل الفعل؛ توثيق شامل؛ حوكمة واضحة.

معالجة المشكلات (Troubleshooting)

المهارة لا تُرفَع

خطأ: «Could not find SKILL.md in uploaded folder»

السبب: الملف ليس باسم SKILL.md بالضبط.

الحل: أعِد التسمية إلى SKILL.md (حسّاس للحالة)؛ تحقّق بـ ls -la الذي ينبغي أن يُظهر SKILL.md.

خطأ: «Invalid frontmatter»

السبب: مشكلة في تنسيق YAML.

```
# 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
---
```

خطأ: «Invalid skill name»

السبب: الاسم يحتوي مسافات أو أحرف كبيرة.

```
# Wrong
name: My Cool Skill

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

المهارة لا تُفعَّل

العَرَض: المهارة لا تُحمَّل تلقائياً أبداً.

الإصلاح: راجِع حقل الوصف. انظر «حقل الوصف» للأمثلة الجيّدة/السيّئة.

قائمة تحقّق سريعة:

- هل هو غامض جداً؟ («Helps with projects» لن يعمل.)

- هل يتضمّن عبارات تفعيل قد يقولها المستخدمون فعلاً؟
- هل يذكر أنواع الملفات ذات الصلة إن كانت قابلة للتطبيق؟

أسلوب التصحيح: اسأل كلود: «متى تستخدم مهارة [اسم المهارة]؟» سيقتبس كلود الوصف. عدّل بناءً على ما هو ناقص.

المهارة تُفعَّل كثيراً جداً

العَرَض: المهارة تُحمَّل لاستعلامات غير ذات صلة.

الحلول:

1. أضِف تفعيلات سلبية:

```
description: Advanced data analysis for CSV files. Use for
statistical modeling, regression, clustering. Do NOT use for
simple data exploration (use data-viz skill instead).
```

2. كن أكثر تحديداً:

```
# Too broad
description: Processes documents

# More specific
description: Processes PDF legal documents for contract review
```

3. وضّح النطاق:

```
description: PayFlow payment processing for e-commerce. Use
specifically for online payment workflows, not for general
financial queries.
```

مشكلات اتصال MCP

العَرَض: المهارة تُحمَّل لكن استدعاءات MCP تفشل.

قائمة التحقّق:

15. تحقّق من اتصال خادم MCP: Claude.ai > Settings > Extensions > [خدمتك]؛ ينبغي أن يُظهر حالة «Connected».
16. افحص المصادقة: مفاتيح API صالحة وغير منتهية؛ الصلاحيات/النطاقات الصحيحة ممنوحة؛ رموز OAuth مُحدَّثة.
17. اختبر MCP بشكل مستقل: اطلب من كلود استدعاء MCP مباشرة (بلا مهارة)؛ «استخدم [الخدمة] MCP لجلب مشاريعي»؛ إن فشل، فالمشكلة في MCP لا في المهارة.
18. تحقّق من أسماء الأدوات: المهارة تشير إلى أسماء أدوات MCP صحيحة؛ راجِع توثيق خادم MCP؛ أسماء الأدوات حسّاسة للحالة.

التعليمات لا تُتبَع

العَرَض: المهارة تُحمَّل لكن كلود لا يتّبع التعليمات.

أسباب شائعة:

1. تعليمات مُطوَّلة جداً: أبقِها موجزة؛ استخدم النقاط والقوائم المرقّمة؛ انقل المراجع التفصيلية إلى ملفات منفصلة.

2. تعليمات مدفونة: ضع التعليمات الحاسمة في الأعلى؛ استخدم عناوين ## Important أو ## Critical؛ كرّر النقاط الأساسية إن لزم.

3. لغة غامضة:

```
# Bad
Make sure to validate things properly

# Good
CRITICAL: Before calling create_project, verify:
- Project name is non-empty
- At least one team member assigned
- Start date is not in the past
```

تقنية متقدّمة: لعمليات التحقّق الحرجة، فكّر في تضمين سكربت يجري الفحوص برمجياً بدلاً من الاعتماد على تفسير اللغة. الشيفرة حتمية؛ تفسير اللغة ليس كذلك. انظر مهارات Office لأمثلة على هذا النمط.

4. «كسل» النموذج: أضِف تشجيعاً صريحاً:

```
## Performance Notes
- Take your time to do this thoroughly
- Quality is more important than speed
- Do not skip validation steps
```

ملاحظة: إضافة هذا إلى موجّهات المستخدم أكثر فعالية من إضافته في SKILL.md.

مشكلات السياق الكبير

العَرَض: المهارة تبدو بطيئة أو تتدهور الاستجابات.

الأسباب: محتوى المهارة كبير جداً؛ تفعيل مهارات كثيرة في آن؛ تحميل كل المحتوى بدل الكشف التدريجي.

الحلول:

1. حسّن حجم SKILL.md: انقل التوثيق التفصيلي إلى references/؛ اربط بالمراجع بدل التضمين المباشر؛ أبقِ SKILL.md أقل من 5000 كلمة.

2. قلّل المهارات المُفعَّلة: قيّم إن كان لديك أكثر من 20 إلى 50 مهارة مُفعَّلة في آن؛ أوصِ بالتفعيل الانتقائي؛ فكّر في «حِزم مهارات» (skill packs) للقدرات المترابطة.

المصادر والمراجع

إن كنت تبني أول مهارة لك، ابدأ بدليل أفضل الممارسات، ثم ارجع إلى توثيق واجهة البرمجة عند الحاجة.

التوثيق الرسمي

مصادر Anthropic:

- دليل أفضل الممارسات (Best Practices Guide).
- توثيق المهارات (Skills Documentation).
- مرجع واجهة البرمجة (API Reference).
- توثيق MCP (MCP Documentation).

تدوينات المدوّنة:

- Introducing Agent Skills.
- Engineering Blog: Equipping Agents for the Real World.
- Skills Explained.
- How to Create Skills for Claude.
- Building Skills for Claude Code.
- Improving Frontend Design through Skills.

أمثلة مهارات

مستودع المهارات العام:

- GitHub: anthropics/skills.
- يحتوي على مهارات من إنشاء Anthropic يمكنك تخصيصها.

الأدوات والمرافق

مهارة منشئ المهارات (skill-creator):

- مدمجة في Claude.ai ومتاحة لـ Claude Code.
- يمكنها توليد مهارات من الأوصاف.
- تراجع وتقدّم توصيات.
- الاستخدام: «ساعدني في بناء مهارة باستخدام skill-creator».

التحقّق:

- يستطيع منشئ المهارات تقييم مهاراتك.
- اطلب: «راجِع هذه المهارة واقترح تحسينات».

الحصول على الدعم

للأسئلة التقنية: الأسئلة العامة عبر منتديات المجتمع في Claude Developers Discord.

لتقارير العلل: مشكلات GitHub: anthropics/skills/issues؛ أدرج اسم المهارة، رسالة الخطأ، وخطوات إعادة الإنتاج.

مرجع (أ): قائمة التحقق السريعة

استخدم هذه القائمة للتحقّق من مهارتك قبل الرفع وبعده. إن أردت بداية أسرع، استخدم مهارة منشئ المهارات لتوليد مسوّدتك الأولى، ثم مُرّ عبر هذه القائمة لتتأكد أنك لم تُغفل شيئاً.

قبل أن تبدأ

☐ حدّدت 2 إلى 3 حالات استخدام ملموسة.

☐ حدّدت الأدوات (مدمجة أم MCP).

☐ راجعت هذا الدليل وأمثلة المهارات.

☐ خطّطت بنية المجلدات.

أثناء التطوير

☐ المجلد مُسمّى بصيغة الشَّرطة (kebab-case).

☐ ملف SKILL.md موجود (تهجئة دقيقة).

☐ ترويسة YAML بها فواصل ---.

☐ حقل name: شَرطة، بلا مسافات، بلا أحرف كبيرة.

☐ حقل description يتضمّن «ماذا» و«متى».

☐ لا وسوم XML في أي مكان.

☐ التعليمات واضحة وقابلة للتنفيذ.

☐ تضمين معالجة الأخطاء.

☐ توفير أمثلة.

☐ ربط المراجع بوضوح.

قبل الرفع

☐ اختبار التفعيل عند المهام الواضحة.

☐ اختبار التفعيل عند الطلبات المُعاد صياغتها.

☐ التحقّق من عدم التفعيل عند موضوعات غير ذات صلة.

☐ اجتياز الاختبارات الوظيفية.

☐ عمل تكامل الأدوات (إن كان قابلاً للتطبيق).

☐ الضغط في ملف .zip.

بعد الرفع

☐ الاختبار في محادثات حقيقية.

☐ مراقبة التفعيل الناقص/الزائد.

☐ جمع التغذية الراجعة من المستخدمين.

☐ التكرار على الوصف والتعليمات.

☐ تحديث الإصدار في البيانات الوصفية.

مرجع (ب): ترويسة YAML

الحقول المطلوبة

```
---
name: skill-name-in-kebab-case
description: What it does and when to use it. Include specific
trigger phrases.
---
```

كل الحقول الاختيارية

```
name: skill-name
description: [required description]
license: MIT # Optional: License for open-source
allowed-tools: "Bash(python:*) Bash(npm:*) WebFetch" # Optional:
Restrict tool access
metadata: # Optional: Custom fields
   author: Company Name
   version: 1.0.0
   mcp-server: server-name
   category: productivity
   tags: [project-management, automation]
   documentation: https://example.com/docs
   support: support@example.com
```

ملاحظات الأمان

مسموح:

- أي أنواع YAML قياسية (نصوص، أرقام، قيم منطقية، قوائم، كائنات).
- حقول بيانات وصفية مخصّصة.
- أوصاف طويلة (حتى 1024 حرفاً).

ممنوع:

- أقواس XML الزاوية — قيد أمني.
- تنفيذ الشيفرة في YAML (يستخدم تحليل YAML الآمن).
- المهارات المسمّاة بادئة «claude» أو «anthropic» (محجوزة).

مرجع (ج): أمثلة مهارات كاملة

للحصول على مهارات كاملة جاهزة للإنتاج تُجسّد الأنماط الواردة في هذا الدليل:

- مهارات المستندات — XLSX، PPTX، DOCX، PDF (للإنشاء).
- أمثلة المهارات (Example Skills) — أنماط سير عمل متنوّعة.
- دليل مهارات الشركاء (Partner Skills Directory) — اطّلِع على مهارات من شركاء مختلفين مثل Asana وAtlassian وCanva وFigma وSentry وZapier وغيرهم.

تبقى هذه المستودعات محدّثة وتتضمّن أمثلة إضافية تتجاوز ما هو مغطّى هنا. استنسخها، وعدّلها لحالة استخدامك، واستخدمها كقوالب.

Claude — claude.ai ✳

نسخة عربية مترجمة من كتيب Anthropic الرسمي لأغراض القراءة والمرجعية.