identifiable
identifiable.ca

Guide Skills Creator

Principes clés pour chaque nouveau Skill

identifiable.ca — Référence opérationnelle

Préparé par Wissam Daibess — identifiable.ca

Avril 2026

01 — AVANT DE COMMENCER

Les 4 questions à répondre avant d'écrire une ligne

Avant de rédiger un seul mot de SKILL.md, valide ces quatre questions. Elles définissent le contrat entre le Skill et son utilisateur.

Question 1 — Qu'est-ce que ce Skill permet de faire ?

Output concret attendu — fichier, code, données, texte structuré.

Question 2 — Quand doit-il se déclencher ?

Phrases triggers, contextes, mots-clés naturels que l'utilisateur utilisera sans y penser.

Question 3 — Quel est le format de sortie attendu ?

Texte, fichier, JSON, .pptx, .docx... Être précis ici évite les interprétations ambiguës.

Question 4 — Faut-il des cas de test vérifiables ?

OUI si l'output est objectivement vérifiable (fichier, code, données). NON si l'output est subjectif (style d'écriture, ton).

02 — STRUCTURE

Anatomie obligatoire d'un SKILL.md

Arborescence de fichiers

```
skill-name/
■■■ SKILL.md              ← Obligatoire (YAML frontmatter + instructions)
■■■ Bundled Resources/    ← Optionnel
    ■■■ scripts/          ← Code exécutable (tâches déterministes)
    ■■■ references/       ← Docs chargées au besoin
    ■■■ assets/           ← Templates, polices, images
```

Le frontmatter YAML — les deux champs obligatoires

```
---
name: nom-du-skill-en-kebab-case
description: >
  Description PUSHY — dire ce que ca fait ET quand déclencher.
  Inclure les mots-clés que l'utilisateur utilisera naturellement.
  Exemples : "Utilise ce skill dès que l'utilisateur mentionne X, Y, Z,
  même s'il ne demande pas explicitement [nom du skill]."
---
```

Règle critique sur la description : Claude a tendance à sous-déclencher les Skills. Rendre la description légèrement pushy — énumérer les phrases triggers explicites.

Système de chargement progressif — 3 niveaux

Niveau	Ce qui se charge	Limite
1 — Metadata	name + description	Toujours en contexte (~100 mots)
2 — SKILL.md body	Instructions complètes	< 500 lignes idéalement
3 — Bundled resources	Scripts, références, assets	Illimité, chargé à la demande

Garder SKILL.md sous 500 lignes. Si on approche la limite, créer des fichiers références et pointer vers eux. Pour les références de plus de 300 lignes, inclure une table des matières.

03 — RÉDACTION

Style d'écriture dans SKILL.md

Forme impérative
Utilise des verbes d'action directs : "Produis…", "Vérifie…", "Génère…" Évite les formulations passives ou conditionnelles.

Expliquer le POURQUOI
Claude comprend mieux l'intent qu'une liste de MUSTs rigides. Expliquer la raison d'une règle vaut mieux que l'imposer sans contexte.

Pas de ALWAYS / NEVER en majuscules
Reformuler en expliquant la raison. Les contraintes absolues sans contexte génèrent des comportements rigides et parfois contre-productifs.

Exemples Before/After
Des exemples concrets avant/après valent plus que des descriptions abstraites. Claude imite mieux ce qu'il voit que ce qu'on lui dit de faire.

Rester général
Le Skill doit fonctionner sur 1000 variantes du même besoin, pas juste sur l'exemple en cours. Évite les instructions trop spécifiques à un cas.

Pattern d'organisation multi-domaines

Quand un Skill couvre plusieurs variantes (ex. présentation / rapport / compte rendu) :

```
skill-name/
■■■ SKILL.md          ← workflow commun + logique de sélection
■■■ references/
    ■■■ presentation.md
    ■■■ rapport.md
    ■■■ compte-rendu.md

# Claude ne charge que le fichier de référence pertinent.
```

04 — TEST & ITÉRATION

Workflow de test et amélioration

Workflow de test — Claude.ai (sans subagents)

1. Écrire le SKILL.md draft
2. Pour chaque cas de test : lire SKILL.md, exécuter la tâche soi-même
3. Présenter le résultat dans la conversation
4. Demander un feedback inline : "Comment c'est ? Qu'est-ce qu'on ajuste ?"
5. Améliorer le SKILL.md selon le feedback
6. Répéter jusqu'à satisfaction

Pas de benchmarking quantitatif en Claude.ai — se concentrer sur l'évaluation qualitative avec l'utilisateur.

Améliorer un Skill existant — 4 principes

Généraliser
Le fix doit fonctionner sur 1000 cas, pas juste le cas testé. Éviter les ajustements trop spécifiques à l'exemple.

Rester lean
Enlever ce qui ne tire pas son poids. Si Claude perd du temps sur une étape, supprimer l'instruction qui la cause.

Expliquer le pourquoi
Transformer les règles rigides en explication d'intent. "Parce que..." fonctionne mieux que "Tu dois absolument..."

Chercher le travail répété
Si Claude réécrit le même helper à chaque run, le bundler dans scripts/ une fois pour toutes.

05 — LIVRAISON & OPTIMISATION

Checklist avant de livrer un Skill

- YAML frontmatter : name + description présents
- Description : pushy, inclut phrases triggers explicites
- SKILL.md : sous 500 lignes
- References : pointées depuis SKILL.md avec guidance sur quand les lire
- Scripts : testés, commentés, dépendances documentées
- Comportement si info manquante : défini explicitement
- Format de sortie : décrit avec exemple concret
- Cas edge case : au moins un scénario "et si…" traité

Optimisation du déclenchement — Claude Code uniquement

```
python -m scripts.run_loop \
  --eval-set trigger_evals.json \
  --skill-path ./skill-name/ \
  --model claude-sonnet-4-5-20251001 \
  --max-iterations 5 \
  --verbose
```

Génère 20 requêtes (10 should-trigger / 10 should-not-trigger). Les meilleures requêtes sont spécifiques, avec contexte, pas abstraites.

BON exemple	MAUVAIS exemple
"mon boss vient de m'envoyer un xlsx avec les ventes Q4 et elle veut une colonne de marge"	"formater des données"

Packaging final

```
python -m scripts.package_skill ./skill-name/

# Produit un fichier .skill installable.
```

Modalités de distribution

Ce guide est un document de référence opérationnelle à usage interne. Pour toute question ou mise à jour, contactez :

Wissam Daibess

Conseil stratégique en IA

identifiable.ca

Laval, Québec