Cours 9

Introduction aux Compétences d'Agent

Anthropic Academy — Resume detaille

Yoann Boulch | 01/04/2026

1. Qu'est-ce que les compétences

■ Les compétences sont des dossiers Markdown réutilisables avec instructions spécialisées

Une compétence est un dossier contenant un fichier SKILL.md avec des instructions détaillées pour une tâche spécifique, découvert et chargé à la demande.

Caractéristiques:

- Réutilisables entre sessions et projets
- Chargées dynamiquement quand nécessaire
- Efficace en contexte: chargent uniquement ce qui est utilité
- Versionnables avec git

2. Emplacements des compétences

■ Compétences personnelles ou par projet

```
# Compétences personnelles (toujours disponibles) ~/.claude/skills/ ■■■ ma-competence/ ■ ■■■ SKILL.md ■■■ autre-competence/ ■■■ SKILL.md # Compétences de projet (ce projet seulement) ./.claude/skills/ ■■■ api-debugging/ ■ ■■■ SKILL.md ■■■ docs-generator/ ■■■ SKILL.md
```

3. Créer votre première compétence

═ Structure: frontmatter YAML + instructions Markdown

```
--- name: "Code Reviewer" description: "Analyse du code pour qualité et sécurité" tools: [read, grep] --- # Rôle Tu es un expert en revue de code... ## Étapes 1. Lire le fichier code 2. Analyser pour: erreurs, sécurité, style 3. Générer un rapport détaillé
```

4. Descriptions efficaces

■ Les descriptions permettent la découverte par matching

Une bonne description inclut mots-clés pertinents et indique clairement ce que fait la compétence.

Conseils pour les descriptions:

- Spécifique: 'Revue de code Python' plutôt que 'Analyse'
- Mots-clés: inclure les termes de recherche courants
- Concis: une phrase, max 10 mots
- Action: verbe clair (analyser, générer, tester)

5. Compétences multi-fichiers

■ Organisez les grandes compétences avec plusieurs fichiers

```
skill-complex/ ■■■ SKILL.md # Instructions principales ■■■ guide.md # Guide détaillé ■■■
examples.md # Exemples d'utilisation ■■■ troubleshooting.md # Dépannage
```

Utilisez la disclosure progressive: SKILL.md contient l'essentiel, les autres fichiers fournissent des détails optionnels.

6. Options de configuration

■ Contrôlez les outils et le comportement

Option	Type	Description
tools	Liste	Outils autorisés pour cette compétence
scripts	Bool	Activer l'exécution de scripts
timeout	Nombre	Timeout en secondes

7. Compétences vs CLAUDE.md vs Slash Commands

■ Choisissez le bon outil pour votre cas d'usage

Outil	Partageabilité	Portée	Cas d'usage
Compétences	Oui (git)	Perso/Projet	Tâches réutilisables
CLAUDE.md	Non	Projet	Configuration locale
Slash Commands	Non	Session	Actions rapides

8. Partager les compétences

■ Distribuez vos compétences via repos ou plugins

Les compétences peuvent être partagées en les commitant dans un repo git public ou en les packagant dans des plugins.

Moyens de distribution:

- Repos public GitHub avec .claude/skills/
- Packages npm avec convention de naming
- Plugins organisationnels (contrôle centralisé)
- Registre communautaire (à venir)

9. Intégration avec les subagents

■ Donnez des compétences spécialisées aux subagents

Les subagents peuvent utiliser des compétences spécifiques pour des tâches délégués.

Modèle subagent + compétence:

1. Agent principal délègue à un subagent spécialisé
2. Subagent charge sa compétence dédiée
3. Compétence fournit instructions et contexte
4. Subagent effectue la tâche avec expertise

Architecture de compétences réutilisables

```
skill-base/ ■■■ SKILL.md # Instructions core ■■■ lib/ ■ ■■■ helpers.py # Code réutilisable ■ ■■■ validators.py # Validation ■■■ examples/ ■■■ usage.md # Exemples
```

10. Versioning et évolution des compétences

■ Gérez les versions et mises à jour

Versionnez vos compétences pour permettre aux utilisateurs de rester sur une version stable.

Bonnes pratiques de versioning:

- Semantic versioning: majeur.mineur.patch
- CHANGELOG: documenter les changements
- Dépréciations graduelles: prévenir les utilisateurs
- Tests de régression: vérifier la compatibilité

11. Mesurer l'efficacité des compétences

■ Analysez l'utilisation et l'impact

Collectez des métriques pour améliorer vos compétences.

Métriques à suivre:

- Taux d'utilisation: combien de gens les utilisent
- Succès: nombre d'exécutions réussies
- Temps d'exécution: performance
- Retours utilisateurs: satisfaction

12. Dépannage des compétences

■ Solutions aux problèmes courants

Problèmes courants et solutions:

- Compétence non découverte: vérifiez la description et les mots-clés
- Erreur de permissions: vérifiez les outils autorisés
- Contexte dépassé: réduisez la taille des fichiers de support
- Conflits de priorité: ajoutez plus de détails à la description

13. Cas d'usage pratiques

■ Exemples concrets et patterns réutilisables

Cas 1: Code Review Skill

Utile pour: revue qualité, sécurité, performance

Outils requis: read, grep pour analyse statique

Cas 2: Documentation Skill

Utile pour: générer docs, README, API specs

Outils requis: read pour analyse, write pour génération

Cas 3: Testing Skill

Utile pour: générer tests, couvrir cas limites

Outils requis: read code, écrire fichiers test

14. Résumé et bonnes pratiques

■ Les compétences sont la clé de la réutilisabilité

Les compétences Claude permettent de créer une base de connaissances réutilisable et maintenable. En suivant les bonnes pratiques de conception, versioning et documentation, vous construirez un écosystème puissant d'expertise distribuée.