

find-skills ב-Claude Code: איך מוצאים, בודקים ומתקינים Skills

הרצתי את find-skills ואת ה-CLI שלו על המחשב שלי: איך מתקינים, איך מחפשים, ומה לבדוק לפני שמתקינים Skill שמצאתם. כולל מה שגיליתי כשהסתכלתי מקרוב על מספרי ההתקנות.

2026 · עודכן ב-11 באוקטובר NexAi - Gilad

הגרסה המעודכנת באתר: nex-ai.co/he/guides/find-skills-claude-code-guide/



הסקיילס find-skills הוא Skill בקוד פתוח של Vercel Labs שמלמד את Claude Code לחפש Skills אחרים באתר `sh.skills` ולהציע אותם להתקנה. הוא לא מעניק ל-Claude הרשאות או כלים שלא היו לו, והוא גם לא מחליף בדיקת איכות ואבטחה שאתם עושים בעצמכם. מה שהוא חוסך הוא החיפוש הידני בין מאגרים.

הפקודות שבמדריך רצו על המחשב שלי ב-11 באוקטובר 2026. ההסברים נשענים על התיעוד של Vercel Labs ושל Claude Code, ומספרי ההתקנות משתנים כל הזמן.

מה זה Skill, ובמה הוא שונה מ-MCP

ה Skill הוא תיקייה עם קובץ `SKILL.md`. בראש הקובץ יש כותרת קצרה עם `name` ו-`description`, ואחריה הוראות שהסוכן קורא כשהמשימה מתאימה. אפשר לצרף לתיקייה גם סקריפטים, תבניות וקבצי עזר. לפי התיעוד של Claude Code, ה-`description` נמצא תמיד בהקשר של Claude והוא מה שקובע מתי ה-Skill נטען. התוכן המלא נטען רק כשהוא נחוץ.

כלומר Skill הוא ידע ותהליך עבודה בכתב. הוא לא מוסיף גישה למערכות. כדי ש-Claude יקרא נתונים מ-Google Sheets צריך חיבור, למשל דרך MCP. כדי שיעבוד על הנתונים האלה בתהליך קבוע, Skill עושה את העבודה. בשפה של [harness engineering](#), ה-Skills הם חלק מהסביבה שמסביב למודל, לצד החוקים וההקשר.

טכנולוגיה	מה היא עושה	איפה היא גרה
Skill	מסביר לסוכן איך לבצע משימה או תהליך	claude/skills/name/SKILL.m. / ~ או d claude/skills/name/SKILL.md. בפרויקט
MCP	מחבר את הסוכן לכלים ולמקורות מידע חיצוניים	מוגדר בחיבורי ה-MCP של Claude Code
find-skills	Skill שעוזר לגלות ולהתקין Skills אחרים	מותקן כמו כל Skill אחר

ה-Skills שאני עובד איתם הם בדיוק זה. humanizer, שמנקה טקסט מדפוסי כתיבה של AI, והסט של hyperframes ליצירת סרטונים, הם תיקיות עם SKILL.md שמותקנות אצלי גלובלית.

מה find-skills עושה בפועל

find-skills עצמו הוא קובץ הוראות. קראתי את ה-SKILL.md שלו, והוא מנחה את Claude לפעול בשישה שלבים:

1. להבין מה המשתמש צריך: תחום, משימה, והאם סביר שכבר קיים Skill לזה.
2. לבדוק קודם את הלוח של skills.sh, שמדרג Skills לפי מספר התקנות.
3. אם הלוח לא מכסה את הצורך, להריץ find skills npx עם מילות מפתח.
4. לאמת איכות לפני שממליצים.
5. להציג למשתמש שם, מספר התקנות, מקור ופקודת התקנה.
6. להתקין רק אם המשתמש רוצה.

בשלב האימות ההוראות קובעות להעדיף Skills עם יותר מ-1K התקנות, להיזהר ממה שמתחת ל-100, להעדיף מקורות רשמיים כמו microsoft-ו-vercel-labs, anthropics, ומהתייחס בחשדנות לריפו עם פחות מ-100 כוכבים ב-GitHub.

זה מה שכתוב בהוראות. מה Claude עושה בפועל תלוי גם בבקשה שלכם, ולכן כדאי לכתוב בה במפורש להשוות בין אפשרויות ולא להתקין כלום לפני אישור.

מה צריך לפני שמתחילים

- Claude Code מותקן.
- Node.js ו-npm, כדי להריץ npx.
- חיבור לאינטרנט.
- הרשאת כתיבה לתיקיות ה-Skills.

ה-CLI הוא החבילה skills ב-npm. בגרסה 1.7.2, שהייתה העדכנית כשבדקתי, ה-package.json מגדיר node 20.0.0 >= אצלי רץ Node 22.15.1, ו-npm הדפיס אזהרת EBADENGINE. הפקודות שהרצתי (חיפוש, רשימה ותצוגה מקדימה) עבדו למרות האזהרה. התקנה על הגרסה הישנה לא בדקתי, ולכן אם אתם מתכוונים להתקין, עדכנו קודם את Node. כך בודקים את הגרסאות:

```
node -v
npm -v
```

אם הפקודות לא מזהות, מתקינים Node.js [מהאתר הרשמי](#).

איך מתקינים את find-skills

1. פותחים טרמינל: PowerShell ב-Windows או Terminal ב-macOS.

2. מריצים את הפקודה:

```
npx skills add https://github.com/vercel-labs/skills --skill find-skills -g -a claude-code
```

3. מוודאים שההתקנה הצליחה:

```
npx skills list -g
```

4. פותחים סשן חדש ב-Claude Code, כדי שה-Skill ייטען.

הדגל -g מתקין לתיקיית המשתמש, כך שה-Skill זמין בכל הפרויקטים. הדגל -a claude-code מצביע על Claude Code כסוכן היעד. בלי הדגלים ה-CLI שואל איזה סוכן ואיזה מיקום. כשהוא מזהה שהוא רץ מתוך סוכן קוד, הוא מדפיס Agent detected ועובר להתקנה בלי שאלות. את זה ראיתי בהרצה שלי.

ההתקנה הגלובלית שומרת את הקבצים ב-./agents/skills/find-skills, וב-./~/.skills/claude/skills נוצר קישור סימלי אליהם. כך זה נראה אצלי. אם אתם מעדיפים עותק במקום קישור, יש דגל --copy.

יש גם צורה קצרה, owner/repo@skill. זו הצורה שהפקודה find מדפיסה בתוצאות:

```
npx skills add vercel-labs/skills@find-skills -g -a claude-code
```

בדקתי עם --list ששתי הצורות מגיעות לאותו Skill.

התקנה לפרויקט או התקנה גלובלית

בלי - g ההתקנה היא לפרויקט הנוכחי. לפי התיעוד, Skills של פרויקט נשמרים יחד איתו, ומי שמשכפל את הריפו מקבל אותם. בריפו של האתר הזה יש תיקיית . agents/skills וקובץ skills-lock.json שמתעד לכל Skill את המקור, הניתב וה-Skills hash. שאני משתמש בהם בכל פרויקט, כמו humanizer, אני מתקין גלובלית. Skills שקשורים לאתר מסוים נשארים בפרויקט.

איך מחפשים Skills

לדבר עם Claude

אחרי ההתקנה אפשר לכתוב בעברית רגילה:

מצא לי Skill שעוזר לכתוב תוכן ולערוך סרטונים. השווה בין האפשרויות, הסבר מה כל אחת עושה, ואל תתקין כלום לפני שאישרתי.

Claude מפעיל את find-skills לפי ה-description שלו, מחפש ומחזיר אפשרויות. את המשפט האחרון בבקשה כדאי לא להשמיט.

לחפש ישירות בטרמינל

פקודת החיפוש היא חלק מה-CLI ופועלת גם בלי find-skills:

```
npx skills find "content writing"
npx skills find playwright
npx skills find react --owner vercel-labs
```

הדגל - owner מגביל את החיפוש למאגרים של בעלים אחד. כך נראו התוצאות הראשונות בשלוש ההרצות ב-11 באוקטובר 2026:

חיפוש	תוצאות ראשונות (התקנות)
content writing	mattpocock/skills@writing-beats (435.9K), mattpocock/skills@writing-shape (435.4K), coreyhaines31/marketingskills@content-strategy (154.7K)
playwright	microsoft/playwright-cli@playwright-cli (180.8K), anthropics/skills@webapp-testing (174.5K), currents-dev/playwright-best-practices-skill@playwright-best-practices (92.1K)

חיפוש	תוצאות ראשונות (התקנות)
react --owner vercel-labs	vercel-labs/agent-skills@vercel-react-best-practices (787.8K), vercel-labs/agent-skills@vercel-composition-patterns (385.1K), vercel-labs/agent-skills@vercel-react-native-skills (237K)

כל תוצאה מגיעה עם קישור לעמוד שלה ב-`skills.sh`. את הטקסט `owner/repo@skill` מדביקים אחרי `npm` `skills add`.

לעיין באתר `skills.sh`

באתר skills.sh יש חיפוש ולוח דירוג. בעמוד של כל Skill מופיעים פקודות ההתקנה, מספר ההתקנות ותוצאות של בדיקות אבטחה חיצוניות. הנה [העמוד של find-skills](#) עצמו.

למה מספר ההתקנות לא מספיק

בעמוד של `find-skills` ב-`skills.sh` הופיעו ב-11 באוקטובר 2026 כ-3.8 מיליון התקנות, 33.7K כוכבים בריפו, ותאריך "first seen" של 26 בינואר 2026. זה מספר של התקנות, ואני לא יודע כמה משתמשים ייחודיים עומדים מאחוריו. ה-CLI שולח טלמטריה עם מזהי ריפו ו-Skill עבור מאגרים ציבוריים (אפשר לכבות עם `DISABLE_TELEMETRY=1` או `DO_NOT_TRACK=1`). אני מניח שהמספרים באתר נשענים על זה, אבל לא מצאתי תיעוד שאומר זאת במפורש.

בשני חיפושים נוספים, `npm skills find automation` ו-`npm skills find "video editing"`, הופיעו בין התוצאות Skills שונים מאותם שישה בעלים, כולם במאגר בשם `superpowers`. מספרי ההתקנות שלהם נעו בין כ-244 אלף לכ-727 אלף. ה-`Skill twitter-automation`, למשל, הופיע תחת כל שישת הבעלים, ובאותו חיפוש הופיע גם Skill בשם זהה מבעלים שביעי, עם 14 התקנות בלבד.

אני לא יודע מה עומד מאחורי המספרים האלה. אפשר להסיק מהם שמספר ההתקנות מעיד על פופולריות ואומר מעט על איכות, ושאותו שם Skill יכול להופיע תחת כמה בעלים. לכן כדאי לבחור לפי המקור, התוכן והבדיקות שבסעיף הבא, ולהתייחס למספר ההתקנות כסימן ראשוני בלבד.

איך בודקים Skill לפני שמתקינים

Skill הוא תוכן שהסוכן קורא ופועל לפיו. מקור לא אמין יכול להכניס הוראות מסוכנות, פקודות הרצה או בקשות לגשת למידע רגיש. הבדיקות שאני ממליץ עליהן:

1. לראות מה יש במאגר בלי להתקין: `npm skills add owner/repo --list`. התיאורים.
2. לקרוא את ה-`SKILL.md` המלא. `npm skills use owner/repo@skill` כותב את קבצי ה-Skill לתיקייה זמנית ומדפיס פרומפט שמכיל את ההוראות המלאות, בלי להתקין אותו אצל הסוכן. הרצתי את זה על `vercel-labs/agent-skills@web-design-guidelines`, והפלט נפתח במשפט שמסביר לסוכן שהוא מקבל Skill לביצוע ואחריו ה-`SKILL.md`. כדאי לשמור את הפלט לקובץ ולקרוא אותו לפני שמעבירים אותו לסוכן.
3. לחפש בקובץ שני דברים שלפי התיעוד של Claude Code משנים את ההרשאות: `allowed-tools`, שמאשר כלים מראש בזמן שה-Skill רץ, והזרקת `shell` עם סימן קריאה ו-`backticks`, שמריצה פקודות לפני ש-Claude רואה את ה-

.Skill

4. לעבור על סקריפטים, קישורים והוראות הרצה שה-Skill מפנה אליהם, ולוודא שאין בהם בקשה לסיסמאות, מפתחות API או קבצים רגישים.

5. להסתכל בעמוד של ה-Skill ב-sh. skills על בדיקות האבטחה. בעמוד של find-skills הופיעו Gen Agent Trust Snyk: Warn ו-Hub: Pass, Socket: Pass. לא בדקתי למה האזהרה של Snyk מתייחסת. מבחינתי Warn הוא סימן לפתוח את הדוח, ולא לדלג עליו.

6. לבדוק את הבעלים של המאגר, את הפעילות בו ואת מספר הכוכבים.

7. להתקין קודם בתיקיית בדיקות ולהריץ משימה קטנה.

זו לא הבטחה לבטיחות. גם תוצאה ירוקה באתר היא בדיקה חיצונית אחת מתוך כמה.

איך מתחזקים Skills אחרי ההתקנה

פעולה	פקודה
רשימת Skills של הפרויקט	<code>npx skills list</code>
רשימת Skills גלובליים	<code>npx skills list -g</code>
עדכון גלובליים בלבד	<code>npx skills update -g</code>
עדכון של הפרויקט בלבד	<code>npx skills update -p</code>
עדכון Skill מסוים	<code>npx skills update find-skills</code>
הסרת Skill גלובלי	<code>npx skills remove -g find-skills</code>
יצירת Skill חדש	<code>npx skills init my-skill</code>

עדכון מושך את הגרסה העדכנית של ההוראות, והוראות הן קוד לכל דבר. ב-Skills ברמת הפרויקט, `git diff` אחרי העדכון מראה בדיוק מה השתנה.

איך יוצרים Skill משלכם

לתהליך שחוזר אצלכם אפשר ליצור Skill משלכם. הפקודה הבאה יוצרת תיקייה עם `SKILL.md` ראשוני:

```
npx skills init website-audit
```

הרצתי אותה בתיקיית בדיקה. הקובץ שנוצר מכיל כותרת עם `name` ו-`description`, וכותרות משנה בשם "When to use" ו-"Instructions". כך הייתי ממלא אותו לביקורת אתרים:

```
---
name: website-audit
description: Audits a website for performance, accessibility, SEO and usability. Use when the user asks to audit
or review a site.
---

# Website audit

1. Inspect the site structure.
2. List accessibility issues.
3. List SEO improvements.
4. Check performance where tools are available.
5. Produce a prioritized action plan.

Never invent measurements. Separate verified findings from recommendations.
```

ה-description חשוב במיוחד, כי לפי התיעוד Claude קורא אותו כדי להחליט מתי להשתמש ב-Skill. תיאורים שמעל 1,536 תווים נחתכים ברשימת ה-Skills, ולכן שמים את מקרה השימוש העיקרי בהתחלה.

כדי ש-Claude Code יטען את התיקיה, שמים אותה ב-`./~/.claude/skills/website-audit` לכל הפרויקטים, או ב-`./~/.claude/skills/website-audit` לפרויקט אחד. ה-CLI תומך גם בהתקנה מנתיב מקומי, `npx skills add website-audit -a claude-code`, אבל את הפקודה הזו לא הרצתי.

במדריך על [סרטוני AI עם Opus 5.5](#) סיפרתי איך אפשר לארוז צינור שלם כ-Skill: בפעם הבאה Claude שואל את הפרטים שהוא צריך ומריץ את התהליך לפי הכללים. על כלים נוספים סביב Claude Code כתבתי ב-[5 כלים ל-Claude Code](#).

התהליך שאני ממליץ עליו

מתחילים בהגדרת משימה מדויקת. "בדיקות אוטומטיות לממשק React לפני פריסה" עובד טוב יותר מ-"AI ל-Skill". אחר כך מבקשים מ-Claude לאתר כמה אפשרויות בלי להתקין אף אחת, עוברים על הבדיקות שבסעיף הקודם, מתקינים אחת ובודקים אותה על משימה קטנה. אם היא שיפרה את התוצאה, משאירים אותה. מדי פעם כדאי לעבור על הרשימה ולהסיר מה שלא בשימוש.

הפרומפט לחיפוש, בניסוח שלי:

השתמש ב-find-skills כדי לחפש Skills שיעזרו לי ב: [המשימה שלי]. הצג עד חמש אפשרויות. לכל אחת תן שם ומאגר מקור, מה היא עושה בפועל, מספר התקנות, כוכבי GitHub ופעילות במאגר, מגבלות וסיכונים אבטחה, ופקודות התקנה מדויקות. דרג לפי התאמה למשימה ולא לפי פופולריות. קרא את ה-SKILL.md ואת הסקריפטים לפני שאתה ממליץ, ואל תתקין או תריץ שום דבר בלי אישור שלי.

שאלות נפוצות

האם find-skills מתקין Skills בלי לשאול אותי?

לפי ה-SKILL.md שלו, הוא מציג אפשרויות ומציע התקנה, ומתקין רק אם המשתמש רוצה. מה שקורה בפועל תלוי גם בבקשה שלכם, ולכן כדאי לכתוב בה "אל תתקין לפני אישור".

האם צריך find-skills כדי להשתמש ב-npx skills find?

לא. `skills find` הוא פקודה של ה-CLI. בתיעוד של ה-CLI אין תלות ב-`find-skills`. Claude נותן ל-Claude את היכולת להפעיל את החיפוש מתוך שיחה.

Claude Code לא מזהה Skill שהתקנתי. מה בודקים?

מריצים `g - skills list` כדי לראות שה-Skill מותקן, ובודקים שנוצר קישור או תיקייה ב-`~/skills/claude`. אחר כך בודקים שה-SKILL.md מתחיל בכותרת YAML תקינה, כי לפי התיעוד, אם ה-YAML לא נקרא, ה-Skill נטען בלי שדות. בסוף פותחים ששן חדש.

האם אפשר להשתמש ב-Skill בלי להתקין אותו?

כן. `npx skills use owner/repo@skill` מדפיס פרומפט עם הוראות ה-Skill בלי להתקין אותו אצל הסוכן. עם הדגל `--agent` הוא מפעיל סוכן נתמך עם הפרומפט.

האם Skills הם בחינם?

ה-CLI ו-`find-skills` הם קוד פתוח. Skills אחרים יכולים להסתמך על שירותים חיצוניים בתשלום, והפעלת סוכן AI עולה לפי המסלול והשימוש שלכם.

מה ההבדל בין Skill ל-MCP?

Skill מלמד את הסוכן איך לבצע תהליך. MCP מחבר אותו לכלי או למקור מידע חיצוני. אפשר לשלב: Skill מגדיר את התהליך, ו-MCP נותן גישה לנתונים.

מקורות

- skills.sh ב-`find-skills`
- מאגר `vercel-labs/skills` ב-GitHub
- תיעוד Skills של Claude Code

לעדכונים, להערכת פרומפטים בלחיצה ולקישורים נוספים, חזרו למדריך באתר.