

איך להוציא את המקסימום מ-Claude Code: לאן הולכים ה-tokens

webinar של Anthropic שמסביר איך session ב-Claude Code צורך tokens (input, output, cache), אילו הרגלים חוסכים, ואילו levers יש ל-admin של צוות

Lydia Hallie (לפי הכיתוב על המסך, השם לא נאמר בהקלטה) · Claude Code workshop של Anthropic, "Maximizing the value of Claude Code" (שותף מחדש ב-X בחשבון Kshitij Mishra) · ספטמבר 2026

משך: כ-27 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה webinar של Anthropic בשם "Maximizing the value of Claude Code" (לפי השקופית, ספטמבר 2026). מציגה אותו מהנדסת מ-Anthropic. לפי הכיתוב על המסך שמה Lydia Hallie, אבל השם לא נאמר בהקלטה עצמה. הנושא הוא איך Claude Code מוציא tokens בפועל: מה נשלח למודל בכל בקשה, למה output יקר מ-input ולמה cache זול מאוד, מה שובר את ה-prompt cache, ואילו הגדרות עומדות לרשות מי שמגדיר את הכלי לצוות שלם.

הטענה המרכזית: יעילות היא לא שימוש בפחות tokens. המטרה היא שה-tokens שכן משלמים עליהם ילכו למשימה שביקשת. ההרצאה מתחילה מהמנגנון ולא מ-checklist, כי לדבריה checklist מתיישן ברגע ש-Anthropic משחררים משהו חדש. הערה חשובה: הכיתוב של הפוסט ב-X ("99% משתמשים ב-Claude Code כמו +100", "Google agents בלולאה", "Chief agent ו-PM agents") לא מופיע בהקלטה בכלל. ההרצאה לא עוסקת ב-swarms של agents.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם timestamps, ומה לא נאמר.
- 2. **המנגנון:** מה נשלח בכל בקשה, שלושת סוגי ה-tokens ומחירון המודלים.
- 3. **מה בשליטתך בתוך session:** בחירת מודל, context רזה ו-effort.
- 4. **ה-cache: prefix, TTL,** מה שובר אותו, ו-subagents.
- 5. **ה-levers לצוות:** managed settings, analytics, OpenTelemetry ומגבלות הוצאה.
- 6-7. **פעולות, שאלות פתוחות ומילון מונחים.**

הערת מקור המסמך נוצר אוטומטית מהתמלול של ההקלטה ומהשקופיות שלה. שמות ומונחים עלולים להשתבש בתמלול האוטומטי (ASR). למשל Claude מתומלל לפעמים כ-cloud ו-cache כ-cash. בכל מקום שבו השקופית והאודיו לא מסכימים, הלכנו לפי השקופית. הציטוטים מועתקים מילה במילה מהתמלול ונבדקו מול ה-timestamp שלהם. לא נוסף שום מידע ממקורות חיצוניים.

1. תקציר מנהלים

ההרצאה בנויה כמודל מנטלי אחד. כל prompt הופך לכמה בקשות. כל בקשה נושאת את כל השיחה עד אותו רגע. ה-cache הופך את ההיסטוריה הזאת לזולה, אבל רק כל עוד תחילת הבקשה (ה-prefix) לא משתנה וממשיכים לעבוד לפני שה-cache פג. כל ההמלצות נגזרות מהמודל הזה, וגם ה-levers ל-admin הם אותן החלטות, רק מוגדרות פעם אחת לכל הצוות.

- **tokens הם היחידה בכל שיטת חיוב:** ב-API ובספקי cloud משלמים לפי token, ובמנויי Pro/Max/Team/Enterprise חלונות ה-5 שעות והחלון השבועי נמדדים גם הם ב-tokens. לדבריה, בעידן ה-[0:00] "we're paying for the work itself" agents.
- **זה אף פעם לא רק ה-prompt:** כל בקשה כוללת system prompt, הגדרות tools, CLAUDE.md, שמות ותיאורים של skills ו-MCP, ורק בסוף את ההודעה שלך. כל tool call מוסיף סבב, וה-context window גדל בכל סבב [3:07].
- **output הוא הכי יקר, cache הוא הכי זול:** את ה-input המודל מעבד בבת אחת, אבל output נוצר token אחרי token, עם מעבר מלא על כל ה-context לכל token. קריאה מה-cache עולה בערך 0.1x ממחיר [4:39–input 6:10].
- **המודל הוא מכפיל של כל השאר:** מודל גדול עושה יותר חישוב לכל token, ב-input וב-output. "Don't run the biggest model on a task that just doesn't need it." [7:43]
- **context רזה חוסך סבבים:** אפשר לציין קובץ ב-@ בתוך ה-prompt, להעביר הוראות ספציפיות ממשימה מ-CLAUDE.md ל-skills, לכבות שרתי MCP שלא בשימוש, ולהריץ test runner במצב שקט [9:14–12:21].
- **session ארוך עולה כפולי:** בדוגמה שלה, שלוש משימות ב-session אחד צרכו פי שניים tokens מאותן שלוש משימות עם clear/ ביניהן [13:54].
- **החלפת מודל או effort באמצע שוברת את ה-cache:** שינוי של token אחד ב-prefix גורם לכך שכל מה שאחריו מחויב במחיר מלא. Fable 5.1 הוא החריג היחיד לגבי שינוי effort [15:28–17:01].
- **ה-cache פג:** כשעה במנוי, בתוך מגבלות השימוש של המנוי, וכחמש דקות ב-API או בספק cloud ול-subagents. כתיבה ל-cache של שעה עולה יותר, אז החלון הארוך משתלם רק למי שעוזב את ה-session ואחר כך חוזר אליו [17:01].
- **ל-admin יש אותם levers, מוגדרים פעם אחת:** דרך managed settings קובעים מודל ברירת מחדל, effort, TTL ושרתי MCP מותרים. דרך OpenTelemetry רואים את שיעור קריאות ה-cache לכל משתמש. הוצאה גבוהה מיוחסת בדרך כלל למודל הגדול שנשאר ברירת מחדל [23:15–21:42].

מה לא נאמר

- **אין agents ואין swarms:** הכיתוב של הפוסט ב-X על "agents +100", Chief agent ו-PM agents לא מופיע בהקלטה. זו הרצאה על חשבונאות tokens, לא על orchestration.
- **אין מספרי חיסכון מצטברים:** מלבד "פי שניים" בדוגמת שלוש המשימות, לא הוצג נתון על כמה חוסך כל צעד בפרויקט אמיתי.
- **המחירון מופיע רק בשקופית:** המחירים לכל מודל לא הוקראו, והמרצה עצמה אומרת שמחירים ופיצ'רים ימשיכו להשתנות.
- **אין ספים מומלצים:** מה נחשב "גבוה" בשיעור ה-cache או בצריכה לכל session. בצילום ה-usage/ בשקופית מופיע 94%, אבל זה לא הוגדר כיעד.
- **אין התייחסות לאיכות:** הטענה ש-effort בינוני על "Fable 5.1" לא נתמכה בנתוני benchmark.
- **ה-Q&A לא נכלל:** ההקלטה נקטעת בסוף ההרצאה, לפני השאלות.

2. המנגנון: מה באמת נשלח למודל

It's never just the prompt

System prompt + built-in tools

CLAUDE.md

Skills · MCP tool names

"fix the failing test in utils.test.ts"

loaded when the session started

← what I typed

CONTEXT WINDOW everything the model gets to see on this request

toward a thing you, you know, ask Claude

"It's never just the prompt": מבנה הבקשה. system prompt ו-built-in tools, CLAUDE.md, שמות של skills ושל MCP tools, ובסוף ההודעה "fix the failing test in utils.test.ts". הכל נטען בתחילת ה-session.

הדוגמה המלווה את ההרצאה: test שנכשל, וה-prompt הוא "fix the failing test in utils.test.ts". Claude Code לא שולח רק את המשפט הזה. הוא שולח bundle שלם שכולל system prompt, הגדרות של tools (read, edit, bash), סביבת עבודה (תיקייה נוכחית ומצב git), CLAUDE.md, ותיאורים קצרים של skills ושל שרתי MCP. המודל מקבל רק שם של קובץ, ולכן התשובה הראשונה שלו היא tool call (Read utils.test.ts) ולא תיקון. ה-harness מבצע את הקריאה ומחזיר את התוצאה, וכך הלאה. לפי השקופיות, prompt אחד הפך לחמש בקשות: Read, Read, Edit, Bash (npm test), ותשובה סופית. ה-context גדל בכל בקשה. כלשונה, "it's never just the prompt". [3:07], ולכן הפס של ה-context window הוא לדבריה הנושא של כל ההרצאה.

The pricing page

	INPUT	CACHED INPUT	OUTPUT
Haiku 4.5	\$1	\$0.10	\$5
Sonnet 5	\$2	\$0.20	\$10
Opus 5	\$5	\$0.50	\$25
Fable 5.1	\$10	\$0.25	\$50

US\$ per million tokens · claude.com/pricing · September 2026

שקופית "The pricing page": מחיר למיליון tokens בדולרים. Haiku 4.5: input \$1, cached \$0.10, output \$5; Sonnet 5: input \$2, cached \$0.20, output \$10; Opus 5: input \$5, cached \$0.50, output \$25; Fable 5.1: input \$10, cached \$0.25, output \$50 (כפי שנקרא מהשקופית).

יש שלושה סוגי tokens, והמחירון בנוי בדיוק לפיהם. Input הוא כל מה שנשלח. Output הוא כל מה שהמודל מחזיר, כולל tool calls, edits, thinking, לא רק הסיכום בסוף. Cached הוא החלק של הבקשה שזהה לבקשה הקודמת. ההסבר למחירים: את ה-input המודל מעבד במעבר אחד, ו-GPU מצטיינים בזה. כל token של output דורש מעבר מלא על כל ה-context. הדוגמה שלה היא $x = 1$: const: מעבר שלם רק כדי לייצר const, ועוד מעבר בשביל x, וכן

הלאה. לכן output הוא הסוג היקר ביותר. בטבלה הכפולה של השקופיות מופיעים גם חלונות השימוש (5 שעות ושבועי) של מנויי Pro, Max, Team, ו-Enterprise, שגם הם נספרים ב-tokens.

Prompt caching

Identical to the previous request: the server keeps what it computed and reads it back at a fraction of the input price

new: computed once, at full price

output

CACHE READ
0.1x
the input price, every request it stays warm

CACHE WRITE · 5-MINUTE
1.25x
paid once, when new content enters

CACHE WRITE · 1-HOUR
2x
the longer window costs more to fill

output is priced the highest of the token

Prompt caching: ה-prefix הזה לבקשה הקודמת נקרא מה-cache בשביר מהמחיר. $0.1 \times$ Cache read ממחיר input. Cache write $1.25 \times$ לחמש דקות $2 \times$ לשעה.

ה-cache הוא לדבריה ה-lever הגדול ביותר. בכל סבב רק מוסיפים לסוף הבקשה, והשרת שומר את ה-state הפנימי שבנה בפעם הקודמת ומחשב רק מהנקודה שבה הבקשה משתנה. החלק הזה בהתחלה נקרא prefix. חלק חדש משלם מחיר מלא פעם אחת, ומהבקשה הבאה הוא כבר בתוך ה-prefix. נכנס הרבה ויוצא מעט, ורוב מה שנכנס נמצא ב-[6:10]: "In a healthy session, most of your input is the cheap cached kind." אפשר לבדוק את זה ב-usage. בצילום המסך בשקופית, שורת ה-Prompt cache מראה 94% מה-input tokens שנקרא מה-cache, במצב warm עם TTL של 5 דקות.

3. מה בשליטתך בתוך session: מודל, context, effort

A smaller model is not a worse model

Haiku	Sonnet	Opus	Fable
quick assistant	generalist	expert	specialist
small scoped jobs subagents	precise edits mechanical changes code in context	subtle bugs unfamiliar territory hard to spell out	the problems nobody else has seen

"A smaller model is not a worse model" Haiku: כעוזר מהיר (משימות מצומצמות, subagents), Sonnet: כ-generalist (עריכות מדויקות, שינויים מכניים, קוד שכבר ב-context), Opus: כמומחה (באגים עדינים, שטח לא מוכר), Fable: כ-specialist (בעיות שאף אחד עוד לא ראה).

השקופית "Four things to optimize" מסדרת את ההמשך: tokens שנכנסים, tokens שיוצאים, כמה מהם ב-cache, ואיזה מודל. המודל בא ראשון כי הוא מכפיל את כל השאר: מודל גדול עושה יותר חישוב על כל token, ב-input וב-[7:43] "a smaller model is not a worse model". output. ל-Sonnet generalist work מספיק.

ובמחיר נמוך בהרבה ל-Opus.token. שווה את התוספת כשיש באג עדין או שטח לא מוכר. ולגבי Fable, בלשונה "Fable is the specialist who's seen the problems no one else has" [7:43].

CLAUDE.md is for what every task needs

1 Task-specific? Make it a skill.
The migration recipe rides along on every request. As a skill it only loads when the model needs it.

2 Still needed at all?

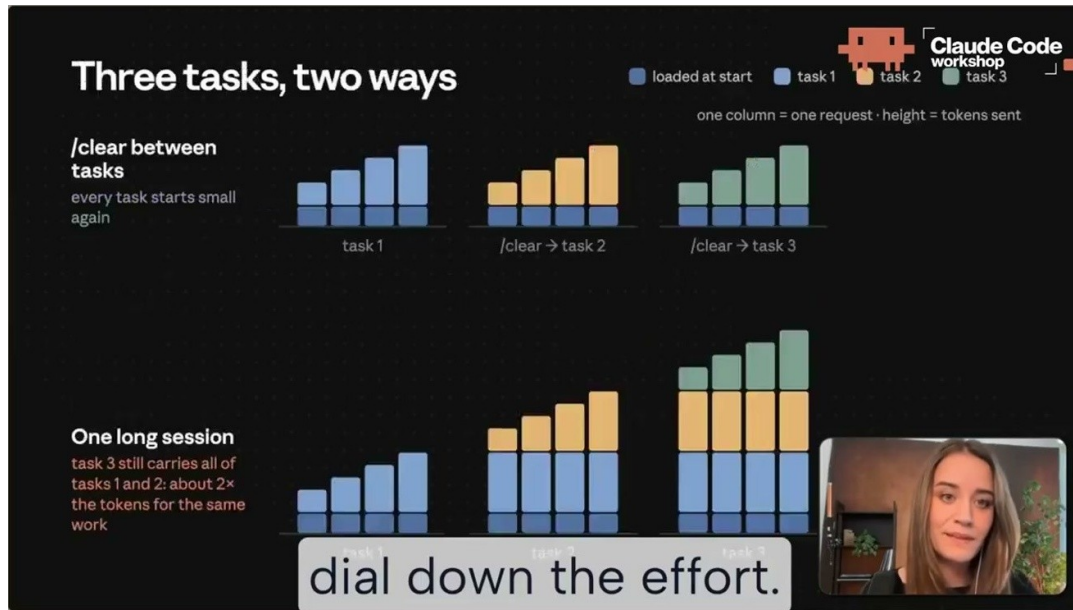
```
# CLAUDE.md
## Project
pnpm monorepo, Node 22
## Commands
- pnpm test --filter <pkg>
- pnpm lint && pnpm typecheck
## Conventions
- named exports, no default exports
## Database migrations
1. pnpm db:generate <name>
2. edit the SQL in /migrations
3. pnpm db:migrate, update schema.md
- explain your plan before editing
```

`> /claude-api prompt-audit`

CLAUDE.md is for what every task needs": דוגמת CLAUDE.md שבה יש מתכון של (1) database migrations הוראה ספציפית למשימה תעבור ל-(2) skill. בדיקה אם ההוראה עדיין נחוצה, בעזרת /claude-api prompt-audit.

- **mention-@ לקובץ:** כשכותבים "utils.test.ts@-ב-prompt Claude Code, מכניס את תוכן הקובץ עוד לפני הבקשה, וה-Read הראשון לא קורה. לפי השקופית זו בקשה אחת במקום שתיים [9:14].
- **context/ בתחילת session:** מראה כמה tokens תופס כל רכיב: system prompt, tools, CLAUDE.md, MCP-ו skills. ההמלצה היא להריץ אותו לפני שמתחילים.
- **CLAUDE.md רק למה שכל משימה צריכה:** "don't keep anything in here that doesn't serve every task" [9:14]. הוראות כמו database migrations עוברות ל-skill, שנטען רק כשהמודל מחליט שהוא נחוץ ועד אז נשלחים רק השם והתיאור שלו.
- **לבדוק אם ההוראות עדיין נחוצות:** מודלים חדשים לא צריכים חלק מההוראות של לפני שלושה חודשים. היא מזכירה skill שבודק את ההוראות ומציע diff. בתמלול השם יצא "Cloud API Prompt Audit", ובשקופית מופיע /claude-api prompt-au... (השם המלא לא ודאי).
- **MCP:** הגדרות tools נטענות היום on demand, ולכן שרת idle זול, אבל לא חינם. אפשר לכבות שרתים ב-/mcp, או להשתמש ב-CLI כמו gh, aws או gcloud, שלא טוען כלום מראש.
- **פלט פקודות נשאר בשיחה:** פלט גדול מאוד נכתב לקובץ ובשיחה נשאר רק preview. אבל test runner שמדפיס 400 שורות יכול להישאר מתחת לסף, ואז כל השורות נכנסות לכל בקשה שאחריו. הפתרון: לכתוב ב-CLAUDE.md את הפקודה בגרסה השקטה, למשל npx vitest run <file> --reporter=dot, או להריץ אותה ב-subagent [12:21].
- **effort:** קובע כמה ביסודיות המודל עובד לפני שהוא מכריז שסיים. הרמות: low, medium, high, xhigh, max. effort גבוה מייצר יותר output וכתוצאה מזה גם יותר input. לדבריה, medium על "excellent" Fable 5.1. כלל אצבע: "did it not know enough or did it not try hard enough?" [13:54]. אם חסר ידע, מחליפים מודל. אם דילג על קובץ או לא הריץ tests, מעלים effort.

4. ה-cache: prefix, TTL, ומה שובר אותו



"Three tasks, two ways": בשורה העליונה /clear בין משימות, וכל משימה מתחילה קטנה. בשורה התחתונה session אחד ארוך, שבו משימה 3 עדיין נושאת את משימות 1 ו-2.

"[13:54] The model only ever adds to your context. It never removes anything." גם כשרוב ה-context ב-cache, זה עדיין לא אפס, ובדוגמה שלה "the bottom session uses twice the amount of tokens as a top" [13:54] session. יש שלושה כלים: compact/ מחליף את השיחה בסיכום, ואפשר לרמוז לו מה חשוב לשמור. rewind/ מוחק את הסבבים האחרונים. /clear פותח שיחה חדשה. אם ייתכן שתצצה לחזור לשיחה הקודמת, תן לה שם קודם ב-rename, ואז resume יחזיר אותה.

המחיר של כל זה: השרת משתמש מחדש רק בחלק שזהה לבקשה הקודמת, החל מה-token הראשון. "But the moment that one token is different, everything after that point is new to the server and gets processed at full price." [15:28] compact ו-clear משנים את ה-prefix בהכרח. אחרי compact זה סיכום קצר, אז המחיר בדרך כלל סביר. החלפת מודל באמצע ה-session מאפסת את ה-cache, כי ה-cache הוא ה-state הפנימי של המודל הספציפי. גם שינוי effort מאפס אותו ברוב המודלים, כי ההגדרה יושבת בתחילת הבקשה. החריג הוא Fable 5.1. Claude Code מבקש אישור לפני /model או /clear effort כשה-cache עדיין warm. לעומת זאת, עריכת קבצים ושינוי permission mode לא נוגעים בבקשה. CLAUDE.md נקרא פעם אחת בתחילת ה-session, ולכן עריכה שלו באמצע תשפיע רק ב-session הבא או אחרי /clear.

What else changes the prefix mid-session

THE NEXT REQUEST AFTER THE CHANGE

> /model

a different model keeps its own cache: nothing of yours is in it yet

all of it read fresh by the new model

> /effort

the effort setting lives inside the front of the request (most models - not Fable 5.1)

read fresh, full price

✓ fine mid-session

editing files, switching permission mode, editing CLAUDE.md (read next session), connecting an MCP server, append at the end

prefix untouched: still read from cache

the change ☒ still cached

conversation so far.

"What else changes the prefix mid-session": /model (למודל אחר יש cache משלו, הכל נקרא מחדש), /effort (ההגדרה בתחילת הבקשה, ברוב המודלים חוץ מ-Fable 5.1), ומה שמותר לשנות: עריכת קבצים, permission mode ועריכת CLAUDE.md.

The cache expires

1h

5m

every cache hit resets the timer

plans, within usage	~1 hour
API - cloud	~5 minutes
subagents	5 minutes

```

~/.claude/settings.json • or managed settings for a team
{
  "promptCacheTtl": "1h",
  "subagentPromptCacheTtl": "1h"
}

```

A 1-hour cache write is billed at 2x input, a 5-minute one at 1x input. Cache expires when you step away and come back, not when you work.

after you clear.

"The cache expires": במנוי, בתוך מגבלות השימוש, כשעה. ב-API או בספק cloud כ-5 דקות. ל-subagents 5 דקות. כל פגיעה ב-cache מאפסת את הטיימר. ההגדרות promptCacheTtl ו-subagentPromptCacheTtl, וכתיבה ל-cache של שעה מחויבת ב-2x.

מה עושים בפועל, לפי השקופית "In practice": בוחרים מודל ו-effort בתחילת ה-session או מיד אחרי /clear. אם צריך להחליף באמצע משימה, עושים compact קודם ורק אז מחליפים, וכך המודל החדש קורא רק סיכום. עושים compact לפני שקמים מהמחשב, כל עוד ה-cache warm, כי compact קורא את כל השיחה פעם נוספת: "try to resume compact or change before you step away, not when you come back." [18:33] את כל השיחה מחדש במחיר input מלא, ולכן כדאי compact לפני שהולכים. ה-TTL הארוך משתלם רק למי שבאמת קם לפגישה או לארוחה וחוזר לאותו session. מי שעובד ברצף משלם עליו סתם.

Subagents: הם רצים ב-context נפרד, עם system prompt קטן, CLAUDE.md והמשימה, ומחזירים לשיחה הראשית רק את המסקנה. במחיר אין חיסכון ישיר: "a subagent isn't free" [20:08]. יש לו input, output ו-cache משלו עם חלון של 5 דקות, והעלות הכוללת של המשימה יכולה אפילו להיות קצת יותר גבוהה. הרווח הוא שהשיחה הראשית נשארת קטנה, וכל סבב אחר כך זול יותר. ל-subagent אפשר לקבוע מודל משלו, למשל Haiku או Sonnet לחיפוש, ו-effort משלו.

5. ה-levers לצוות: managed settings ו-observability

Managed settings מ-admin console: קובץ אחד שכל התקנה טוענת ב-startup, ומשתמשים לא יכולים לדרוס אותו. בדוגמה: model: "sonnet", availableModels: ["sonnet", "haiku", "opus"], effortLevel: "medium", maxEffortLevel: "high", promptCacheTtl: "1h", allowedMcpServers: [...], env: { "CLAUDE_CODE_ENABLE_TELEMETRY": "1", ... }

מי שמגדיר את Claude Code לצוות מקבל את אותם levers, פעם אחת לכולם, דרך managed settings. אפשר לקבוע את מודל ברירת המחדל ואילו מודלים מותרים בכלל. לדבריה זה ה-lever החזק ביותר: "when a team's spend is unexpectedly high, this usually traces back to just the biggest model, just being left on as a default." [21:42] אפשר לקבוע effort התחלתי (משתמש עדיין יכול להעלות אותו למשימה מסוימת), וב-Enterprise גם תקרת effort לכל מודל. אפשר לקבוע TTL ל-cache לצוותים שעובדים דרך API או cloud, רשימת שרתי MCP מותרים, ו-CLAUDE.md ארגוני שמגיע לכל מכונה.

יש גם כלים לראות מה קורה בפועל. ב-console יש analytics של אימוץ ודוח הוצאה לפי משתמש ולפי מודל, כשהוצאה דרך usage credits מופעלת. OpenTelemetry מופעל בשני משתני סביבה באותו קובץ, ומדווח input, output, cache reads ו-cache writes לכל אדם. שיעור cache נמוך אצל משתמש מעיד שהוא שובר את ה-cache, כנראה בהחלפת מודל או effort. קו של input לכל session אמור לעלות במהלך משימה ולרדת אחרי clear. "If it only ever climbs, then someone is just living in one endless session" [23:15] מגבלות: חלונות ה-5 שעות והשבועי של ה-seat הם התקרה. אפשר להפעיל usage credits עם spend limit לארגון, לקבוצה או לאדם, ו-autoContinueAtUsageLimit קובע אם session שנעצר במגבלה ימשיך לבד כשהחלון מתאפס. זה חשוב לריצות ארוכות בלי השגחה.

OpenTelemetry

```

"env": {
  "CLAUDE_CODE_ENABLE_TELEMETRY": "1",
  "OTEL_METRICS_EXPORTER": "otlp",
  "OTEL_EXPORTER_OTLP_ENDPOINT": "https://collector:4317"
}

```

Cache reads, as a share of each person's input

Developer	Share
dev A	92%
dev B	88%
dev C	41%

should be by far the biggest chunk; a low share means the cache keeps breaking; model or effort switches, long gaps

Input per request, over a session

DROPS AT /CLEAR **ONLY CLIMBS**

a climb that never resets is one endless session, or a resume after the cache expired

Cost by model, per person

Developer	Model
dev A	mostly Sonnet
dev B	mostly Sonnet
dev C	all Opus

shows who has the biggest model on for everything; each of these is a one message fix

credits.

OpenTelemetry: המשתנים `OTEL_METRICS_EXPORTER="otlp"`, `CLAUDE_CODE_ENABLE_TELEMETRY` ו-`OTEL_EXPORTER_OTLP_ENDPOINT`, ושלושה גרפים: `cache reads` כשיעור מה-`input` לכל משתמש, `input` לכל בקשה לאורך `session`, ועלויות לפי מודל לכל משתמש.

6. מה לעשות מחר

Three things to take home

1

Decide at the start, then leave it

model · effort · MCP servers

2

Be deliberate about what goes in

@ the files you know · a lean CLAUDE.md · task-specific steps in skills · noisy work in a subagent

3

Don't let one session run forever

/clear for a new task · /compact before you step away

The model multiplies all of it. For a team: the same choices, as managed settings.

cache stays the same and you keep

"Three things to take home": (1) להחליט בהתחלה ולא לגעת (`model`, `effort`, `MCP`). (2) להיות מכוון לגבי מה נכנס (@ לקבצים, `CLAUDE.md` רזה, `skills`, פלט שקט, `subagents` לחיפוש רועשים). (3) לא לתת ל-`session` לרוץ לנצח (`clear` למשימה חדשה, `compact` לפני הפסקה).

- **היום, ב-`session` שלך:** להריץ `context` ב-`session` חדש ו-`usage` ב-`session` פעיל, ולבדוק ששיעור ה-`cache` גבוה.
- **לנקות את `CLAUDE.md`:** להעביר הוראות שרלוונטיות רק למשימות מסוימות ל-`skills`, ולמחוק הוראות שמודלים חדשים כבר לא צריכים.
- **להוסיף ל-`CLAUDE.md` את הפקודות השקטות:** שתיים-שלוש הפקודות שמריצים כל יום, כתובות בדיוק כמו שמריצים אותן, עם `reporter` שמדפיס רק כשלונוות. אחרי `commit` זה תקף לכל הצוות.

- **לבחור מודל ו-effort בתחילת ה-session:** Sonnet ו-medium כנקודת פתיחה. להחליף רק אחרי compact או clear.
- **הרגלי הפסקה:** compact לפני פגישה או לפני סוף היום, לא אחריהם. rename ואז clear כשעוברים נושא.
- **ל-admin:** לבדוק מה מודל ברירת המחדל ב-managed settings, להפעיל OpenTelemetry, ולעקוב אחרי שיעור ה-cache וצורת הקו של input לכל session.

שאלה פתוחה	למה זה חשוב
כמה חוסכים בפועל @-mention, פלט שקט ו-CLAUDE.md רזה בפרויקט אמיתי?	הוצגה רק דוגמת "פי שניים" של clear. אין נתון שמאפשר לתעדף.
מה נחשב שיעור cache "בריא"?	94% מופיע רק בצילום מסך. אין סף שאפשר להגדיר עליו התראה ב-OpenTelemetry.
למה Fable 5.1 לא שובר cache בשינוי effort, והאם זה יחול על מודלים נוספים?	זה משנה את ההמלצה "לא לגעת ב-effort באמצע".
מתי TTL של שעה משתלם, בהינתן cache write של 2x מול 1.25x?	נדרש חישוב break-even לפי דפוס העבודה של הצוות.
האם effort medium באמת לא פוגע באיכות במשימות מורכבות?	הטענה נאמרה בלי benchmark.

7. מילון מונחים

מונח	משמעות
token	יחידת טקסט קטנה, בערך בגודל מילה. היחידה שלפיה מחויבים ונמדדות המגבלות.
context window	כל מה שהמודל רואה בבקשה אחת. גדל בכל tool call.
input / output / cached tokens	מה שנשלח למודל, מה שהמודל מייצר (כולל tool calls ו-thinking), והחלק שנקרא מה-cache במחיר מופחת.
prompt cache / prefix	שמירת ה-state של החלק הזהה בתחילת הבקשה. שינוי של token אחד בו מבטל את כל מה שאחריו.
TTL	כמה זמן ה-cache נשמר. כשעה במנוי, כחמש דקות ב-API ול-subagents.
harness	Claude Code עצמו, שמבצע את ה-tool calls שהמודל מבקש.
effort	כמה ביסודיות המודל עובד לפני שהוא מסיים: low, medium, high, xhigh, max.
CLAUDE.md	קובץ הוראות שנשלח בכל בקשה. נקרא פעם אחת בתחילת session.
skill	הוראות שנטענות רק כשהמודל מחליט שהן נחוצות. עד אז נשלחים רק השם והתיאור.
MCP server	שרת שחושף tools למודל. גם כש-idle הוא עולה מעט tokens.
subagent	agent שרץ ב-context נפרד ומחזיר לשיחה הראשית רק מסקנה.
compact, /rewind, /clear/	סיכום השיחה, מחיקת הסבבים האחרונים, ופתיחת שיחה חדשה.
managed settings	קובץ הגדרות ארגוני מה-admin console שמשתמשים לא יכולים לדרוס.
OpenTelemetry	תקן לייצוא metrics. משמש לדיווח צריכת tokens לפי משתמש.
usage credits / spend limit	הוצאה מעבר לחלון של ה-seat, עם תקרה לארגון, לקבוצה או לאדם.