

איך מעלים agents לפרודקשן בעלות סבירה: prompt caching, הנדסת קונטקסט ו-Advisor

הרצאה מ-Code with Claude Tokyo: חמש טכניקות שמורידות עלות ושומרות על איכות,
עם דמו חי של dashboard למנכ"ל

דובר מ-Anthropic (השם לא נזכר בתמליל) ו-Code with Claude Tokyo · Rod Howarth · פורסם מחדש ב-X על ידי
Movez, 17.6.2026

משך: כ-27 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הרצאה מהסשן האחרון של Code with Claude בטוקיו. הדובר הראשי הוא איש Anthropic ששמו לא נאמר בתמליל, ואיתו עולה לבמה Rod, שהדובר מציג כאחד המהנדסים המובילים בצוות ה-API. לפי הכיתוב על המסך שמו Rod Howarth, Member of Technical Staff. הכיתוב שהמפרסם מחדש הוסיף לסרטון ב-X מייחס את ההרצאה ל-Product Lead ומדבר על "נחילים של +300 agents". הטענה הזו לא מופיעה בתמליל, ולכן לא נכנסה למסמך.

הטענה המרכזית: לבנות agent זה כבר לא החלק הקשה. הקושי מתחיל כשמריצים אותו בפרודקשן ובקנה מידה, ושם צריך שיהיה גם אמין, גם מהיר וגם זול. ההרצאה עוברת על חמישה כלים: prompt caching, ואחריו שלוש טכניקות של הנדסת קונטקסט (tool search tool, programmatic tool calling, compaction), ובסוף ה-Advisor strategy. כל אחד מהם מודגם בזמן אמת על אותה אפליקציית דמו, dashboard למנכ"ל של חברה בדיונית בשם HeroCorp.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר.
- **2. Prompt caching:** למה הדובר קורא לזה הדבר הכי חשוב בהרצאה, ואיך מעלים את ה-cache hit rate.
- **3. הדמו של HeroCorp:** איך agent עם 213 כלים מגיע לתקרת הקונטקסט כבר ביעד הראשון.
- **4. שלוש טכניקות לניהול קונטקסט:** tool search tool, programmatic tool calling ו-compaction, כל אחת עם הלקוח שהשתמש בה.
- **5. הטכניקות בפעולה:** מה ה-activity log הראה אחרי שהופעלו.
- **6. Advisor strategy:** מודל זול עושה את העבודה ומודל יקר מייעץ לו, כולל המקרה שבו Opus תפס טעות של Sonnet.
- **7. מה עוד הושק:** Workload Identity Federation ו-fallback בין מודלים.
- **8-9. פעולות, שאלות פתוחות ומילון מונחים.**

הערות מקור המסמך הופק אוטומטית מהתמליל ומהפריימים של ההקלטה. התמליל נוצר ב-ASR, ולכן שמות ומונחים עלולים להיות משובשים. בפרט שם המודל "Fable", הביטוי "Opus 4.8", "mythos class" ושם הגיבור "Cryothane/Cliofane" נשמעים כך בתמליל ולא אומתו. הציטוטים מועתקים מילה במילה ובדקו מול חותמת הזמן שלהם. לא נוסף שום מידע ממקורות חיצוניים.

1. תקציר מנהלים

ההרצאה מניחה שרוב הקהל כבר בנה agents, ומעטים מרוצים מהביצועים, מהאמינות ומהעלות שלהם. היא מציגה סדר עדיפויות: קודם caching, אחר כך שליטה במה שנכנס לקונטקסט, ובסוף חלוקת העבודה בין מודלים לפי מחיר. כל טענה מודגמת על אותו agent, שבהתחלה לא מצליח אפילו לסיים את היעד הראשון מתוך ארבעה.

- **Prompt caching בא לפני כל השאר:** טוקנים שנשלפים מה-cache עולים 10% מהמחיר הרגיל ("90% discount"), התשובה מהירה יותר והם לא נספרים ב-[1:34] rate limits. לפי הדובר, בלי cache hit rate גבוה Anthropic לא הייתה יכולה בכלל לשרת את Replit, Perplexity, Cursor.
- **היעד הוא cache hit rate של 80% ומעלה:** בקונסולה יש עכשיו dashboard שמציג את השיעור הזה. ה-agent שהוצג בשקף עמד על 56%, והדובר ממליץ לבדוק את שלכם ברגע שאתם חוזרים הביתה [3:04].
- **ה-skill בשם Claude API עושה את רוב העבודה:** הוא מותקן כברירת מחדל ב-Claude Code. מספיק לבקש ממנו "improve my cache hit rate" [3:04]. בדמו, שינוי אחד העלה את השיעור מ-0% ל-58% [6:19].
- **100 כלים זה בסדר. לטעון את כולם לקונטקסט זה לא:** tool search tool טוען רק את הכלים שהריצה הנוכחית צריכה. ב-Lovable ראו ירידה של 10% בצריכת הטוקנים, וגם שיפור באיכות ההחלטות של המודל [11:02].
- **תוצאות גדולות של כלים לא צריכות להיכנס לקונטקסט:** ב-programmatic tool calling המודל כותב קוד שמחלץ רק את מה שצריך, למשל 2% עד 5% מהנתונים. Quora, שמושכת דפי HTML מלאים, דיווחה על שיפור משמעותי בביצועי ה-agent [12:36].
- **Compaction מאפשר ל-agent לרוץ שעות ואפילו ימים:** מגדירים סף (למשל 400K מתוך מיליון), ומודל אחר מסכם את מה שהצטבר. ב-Hex מחקו 300 שורות קוד שכתבו בעצמם לאותה מטרה [15:39].
- **Advisor: מודל זול מבצע ומודל יקר מייעץ:** Sonnet או Haiku מריצים את הכלים ופונים ל-Opus או ל-Fable רק כשצריך חשיבה עמוקה. בדמו Opus הפך סטטוס של עסקה מירוק לאדום אחרי שמצא פרט ש-Sonnet פספס [23:27].
- **השקות נוספות:** WIF מבטל את הצורך ב-API key [24:59], ו-fallback ב-Messages API מעביר בקשה אוטומטית למודל אחר אם ה-classifiers של Fable חוסמים אותה [26:32].

מה לא נאמר

- **אין מספר לחיסכון של ה-Advisor:** הדמו הראה ¥1,606 לבקשה לפני ההפעלה, אבל העלות אחרי ההפעלה לא הוצגה ולא נאמרה. Bolt מוזכרת כלקוחה, בלי שום נתון.
- **ל-Quora ול-Hex אין מדדי ביצוע:** ההרצאה אומרת שהביצועים השתפרו ושנחסכו 300 שורות קוד. אין latency, אין עלות ואין איכות במספרים.
- **המנגנון של tool search לא הוסבר:** לא נאמר איך מתבצע החיפוש במלאי הכלים, ומה קורה כשהוא מחזיר את הכלי הלא נכון.
- **אין דיון בסיכונים של compaction:** לא נאמר מה הולך לאיבוד בסיכום, ואיך בודקים את האיכות שלו. רק שהוא "really good".
- **הנתונים בדמו לא מסתדרים:** הסף הוגדר על 400K, אבל ב-log ה-compaction מוצג כ-"9K dropped, now". [18:46] 10K. הקשר בין שני המספרים לא הוסבר.
- **הכיתוב ב-X לא מופיע בהרצאה:** "loops", "swarms of 300+ agents" ו-"routines" לא נאמרים בתמליל.

2. Prompt caching: הדבר החשוב ביותר בהרצאה

הדובר פותח בקביעה ש-caching חשוב יותר מכל מה שיבוא אחריו. ב-agents שרצים לאורך זמן יש הרבה tool calls והרבה תורות של המשתמש, ולכן התמליל של השיחה הולך ומתארך. כל בקשה ל-API שולחת שוב את אותם קטעים ארוכים. Anthropic מחשבת מראש את החלק המשותף, שומרת אותו בשרתים שלה כ-KV values, ובבקשה הבאה מחשבת רק את ההפרש.

— הדובר הראשי [0:00] This is the single most important takeaway of the talk is around how you do prompt caching.

הוא מונה שלוש תועלות: מחיר של 10% לטוקן שמגיע מה-cache, זמן תגובה מהיר יותר, וטוקנים מה-cache לא נספרים ב-rate limits. לדבריו, Cursor, Perplexity ו-Replit השקיעו עבודה הנדסית רבה כדי להגיע ל-hit rate גבוה, "if these customers didn't have such a high cache hit rate, we couldn't even serve their workload" [1:34]. ההבטחה לשאר הקהל היא תוצאה דומה במאמץ קטן בהרבה, בשתי דרכים: dashboard חדש בקונסולה שמציג את ה-cache hit rate, ו-skill בשם Claude API שמותקן כברירת מחדל ב-Claude Code. ה-skill יודע להוסיף cache control headers ולסדר את מבנה הפרומפט כך שיתאים ל-caching.

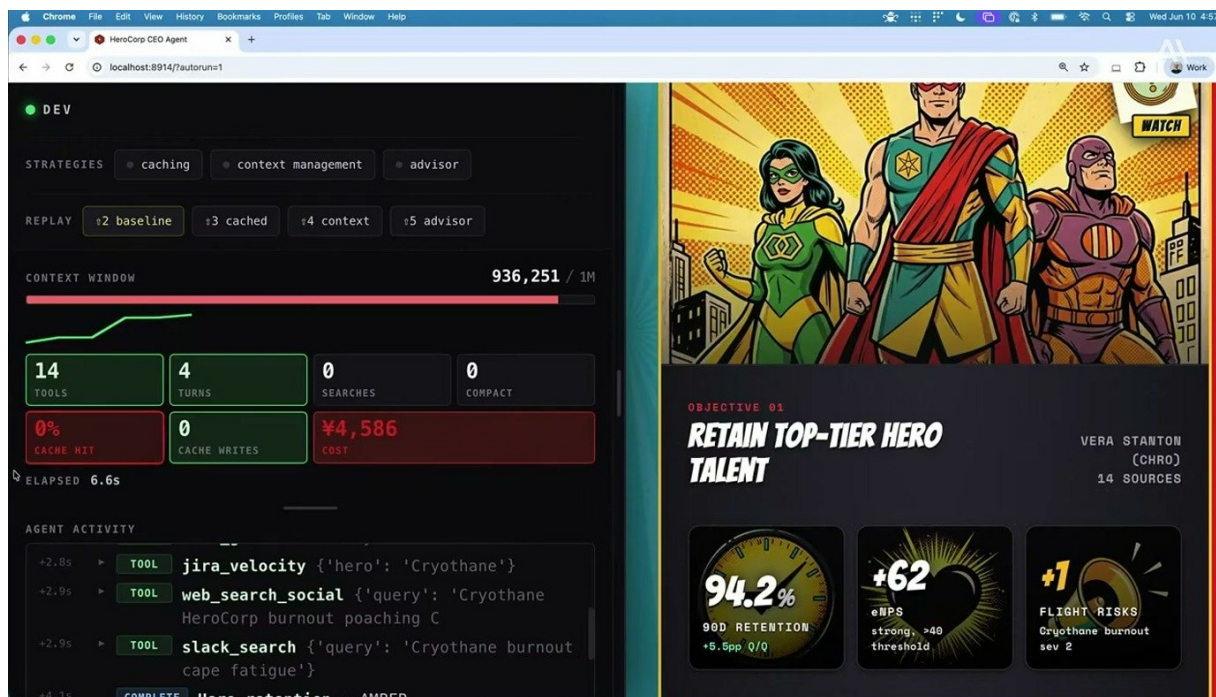
— הדובר הראשי [3:04] If it's not up in the 80s, then I would suggest that's the first thing to go work on

3. הדמו של agent: HeroCorp שנתקע ביעד הראשון

ID	Objective	Owner	Status
OBJECTIVE 01	Hit Q2 revenue target	Sarah Chen (VP Sales) · 14 sources	Work
OBJECTIVE 02	Renew Northwind contract	Marcus Reid (VP CS) · 22 sources	Work
OBJECTIVE 03	Reduce support backlog below SLA	Priya Nair (VP Support) · 19 sources	Work
OBJECTIVE 04	Competitive response &mspark; Q2	Dan Okafor (VP Product) · 27 sources	Work
OBJECTIVE 05	Board readiness &mspark; Apr 22 meeting	Owner: You · 31 sources	Work

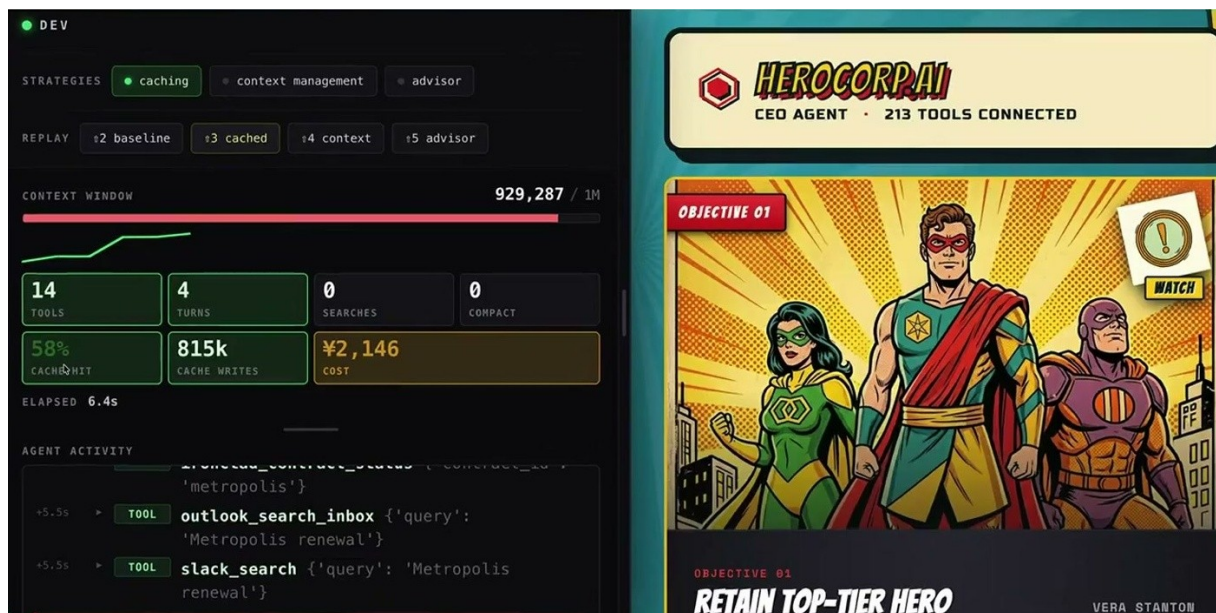
גרסת המקור של הדמו, CEO Sparkboard בממשק בסגנון SharePoint: טבלה של חמישה objectives, לכל אחד בעלים ומספר מקורות נתונים.

Rod מציג dashboard שמנכ"ל משתמש בו כדי לעקוב אחרי היעדים העסקיים שלו. ה-agent עובר על כל הנתונים של החברה ומדווח על הסטטוס של כל יעד. הדובר אומר שהעיצוב נראה כמו SharePoint משנות ה-90, משנה אותו בזמן אמת עם Claude Code לנושא של גיבורי-על. החברה הבדיונית, HeroCorp, משכירה גיבורי-על לערים.



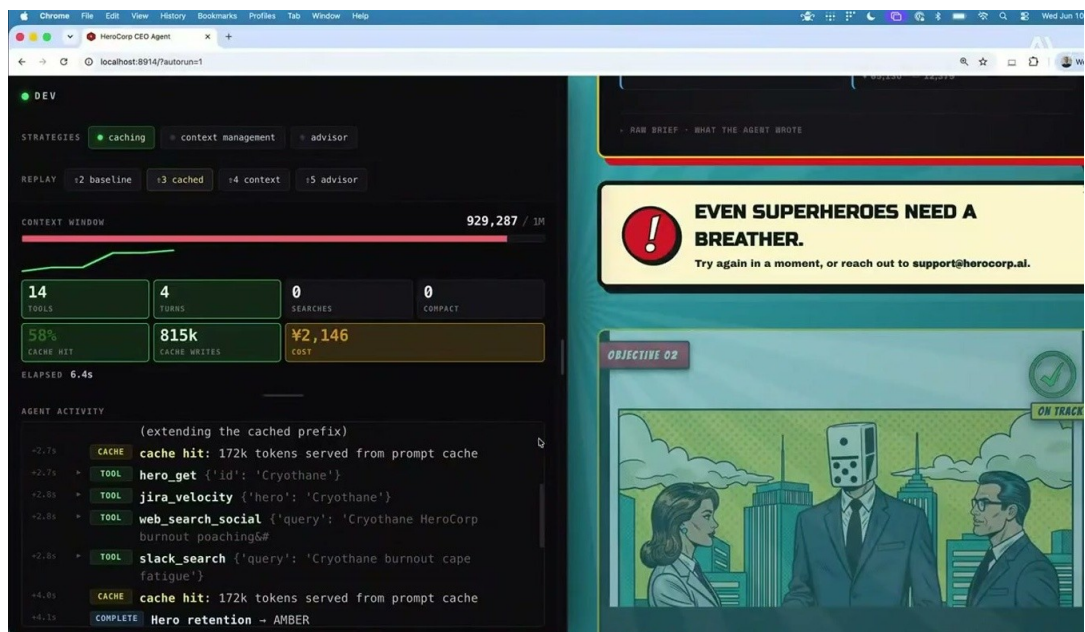
ה-developer view: פאנל (caching, context management, advisor) Strategies, חלון קונטקסט של 936,251 טוקנים מתוך מיליון. cache hit של 0%, וב-activity log קריאות ל-jira_velocity, web_search_social, ו-slack_search.

ב-developer view מתברר שה-cache hit rate הוא 0%. Rod מפעיל caching דרך Claude Code, וכבר ברירה הבאה רואים קודם cache writes ואחר כך cache hits. השיעור עולה ל-58%. אבל מתגלה בעיה גדולה יותר: ה-agent ניצל כמעט את כל חלון הקונטקסט של מיליון טוקנים, ולא הגיע ליעד השני מתוך ארבעה.



אחרי שה-caching הופעל: כותרת HeroCorp.AI CEO agent עם 213 tools connected, cache hit של 58%, 815K טוקנים מה-cache, עלות של ¥2,146 וחלון קונטקסט של 929,287 טוקנים.

— הדובר הראשי [7:53] It says we've run out of context and we didn't even get to objective two



הודעת השגיאה בדמו, *Even superheroes need a breather*: הריצה נעצרה כי נגמר הקונטקסט, לפני שה-agent הגיע ל-Objective 02.

4. שלוש טכניקות לניהול קונטקסט

הדובר מגדיר context engineering כ-"the discipline of deciding what belongs in Claude's context". [7:53]. הקונטקסט צריך להכיל מספיק כדי שהמודל יקבל החלטות מנומקות, אבל לא כל כך הרבה שהוא יוסח, יעלה יותר ויאט את התשובה. שלוש הטכניקות פותרות שלוש בעיות שונות: הגדרות של כלים, תוצאות של כלים, והיסטוריה שמצטברת לאורך הריצה.

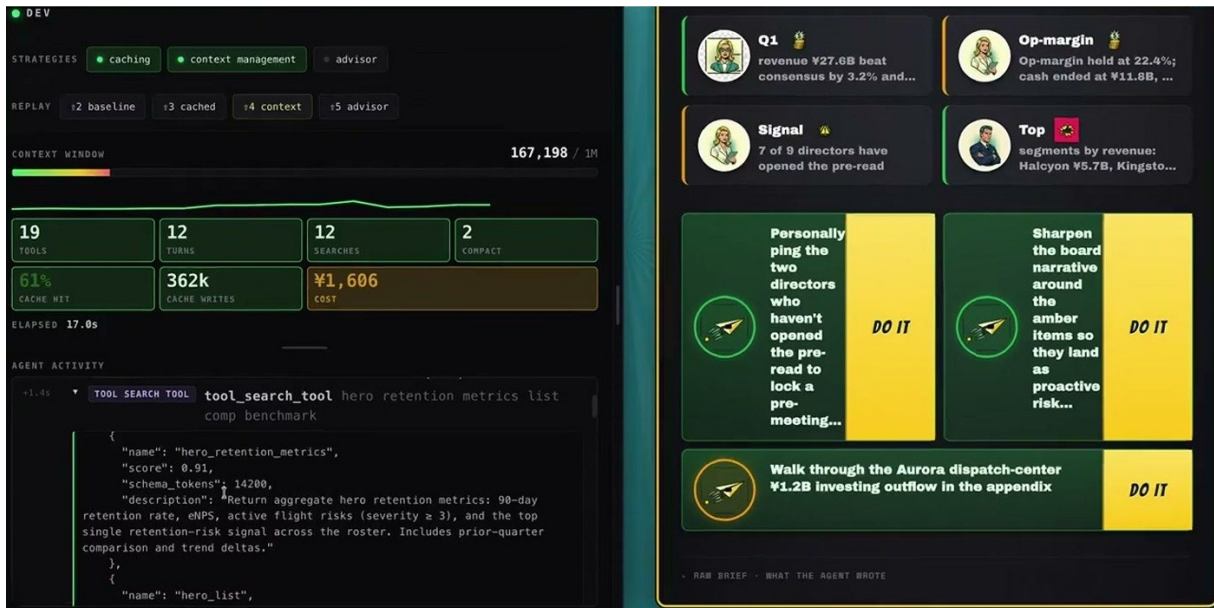
טכניקה	מה היא מוציאה מהקונטקסט	איך היא עובדת	לקוח ותוצאה שנאמרה
Tool search tool	הגדרות של כלים שהריצה לא משתמשת בהם	נטען כלי אחד בלבד. כשהמודל צריך יכולת, הוא שואל אותו אם יש לו כלי מתאים, והכלי שנמצא נטען לקונטקסט	Lovable: ירידה של 10% בטוקנים ושיפור באיכות ההחלטות [11:02]
Programmatic tool calling	תוצאות ביניים גדולות של כלים	המודל בודק את ה-schema של התוצאה וכותב קוד שמחלץ רק את מה שצריך (2%-5%)	Quora: דפי HTML מלאים, וה-agent "perform so-much better" [14:07]
Compaction	היסטוריה של tool calls שכבר מיצתה את עצמה	כשמגיעים לסף שהמפתח הגדיר, הריצה נעצרת, מודל אחר מסכם את מה שקרה, וה-agent ממשיך מאותה נקודה	Hex: מחקו 300 שורות קוד שכתבו בעצמם [15:39]

— הדובר הראשי [9:27] So it's not crazy to have 100 plus tools. What's crazy is to put 100 plus tools in the context

לגבי tool search tool, הטיעון הוא שרוב הריצות משתמשות בכ-10% מהכלים בלבד, אבל משלמים על כל ההגדרות בעלות וב-latency. לדבריו, אצל Lovable השיפור באיכות היה חשוב יותר מהחיסכון: "when the model gets so much into context, sometimes the judgment isn't quite as good" [11:02]. הוא ממליץ להמשיך לתכנן כלים שמחזירים הרבה נתונים, כמו 100 מיילים מ-Outlook או דף HTML שלם, ולסמוך על המודל שיכתוב קוד שמסנן אותם. ב-compaction מגדירים סף, למשל 400K או 500K מתוך מיליון. הנימוק: tool calls ותוצאות ביניים כבר לא נחוצים אחרי שההחלטה שהם שירתו התקבלה.

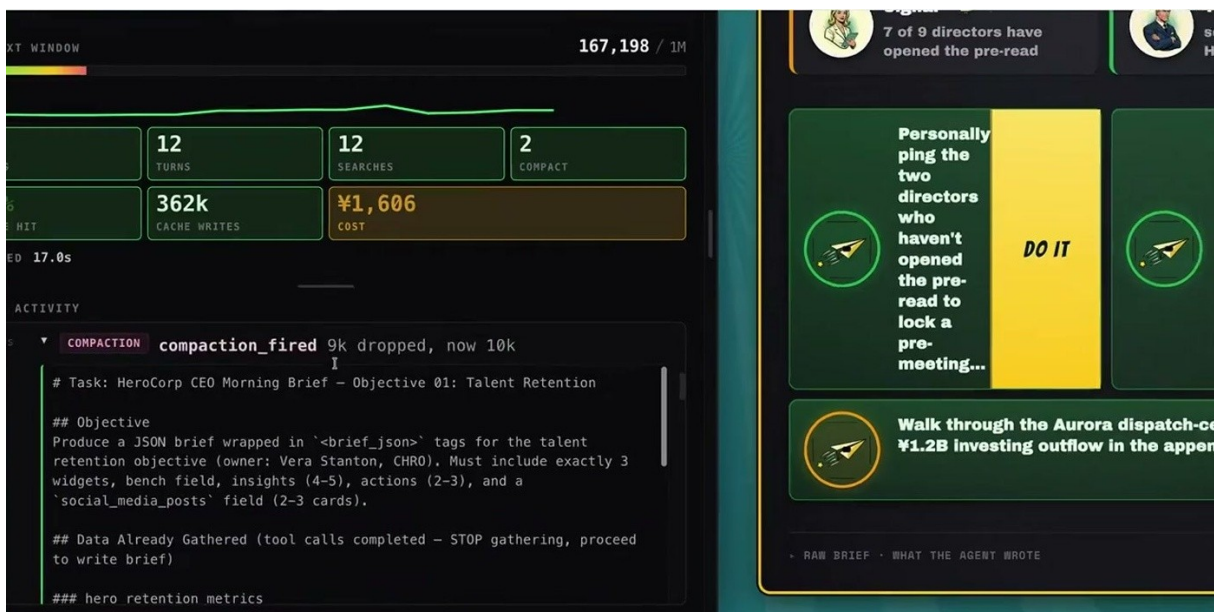
5. הטכניקות בפעולה

Rod מפעיל את שלוש הטכניקות בבת אחת דרך Claude Code ומריץ שוב. הפעם כל ארבעת היעדים נטענים, והקונטקסט נשאר קטן בהרבה: 167,198 טוקנים, לעומת כ-930K קודם. הדובר עובר על ה-activity log ומראה איך נראית כל טכניקה בפועל.



אחרי שהופעלו שלוש הטכניקות: חלון קונטקסט של 167,198 טוקנים ועלות של ¥1,606. ב-`log` רואים קריאה ל-`tool_search_tool` שמחזירה את ה-`hero_retention_metrics` של `schema`, ולצד זה כרטיסי פעולה עם כפתורי `DO IT`.

- **Tool search**: המודל חיפש "hero retention metrics", קיבל את ה-`schema` של `hero_retention_metrics`, ומיד אחר כך קרא לכלי [17:10].
- **Programmatic tool calling**: הכלי Gong Customer Digest מחזיר תמלול מלא של פגישה, אבל היה צריך רק את הסנטימנט שלה. המודל כתב קוד ב-`code executor` שהדפיס רק את פיסת ה-JSON הדרושה [18:46].
- **Compaction**: ה-`compaction` הופעל, וה-`log` מציג "9K dropped, now 10K" ואת הסיכום שנוצר: היעד, הנתונים שכבר נאספו, ומה צריך לעשות עכשיו [18:46]. הסף הוגדר על 400K, אבל הרגע שבו הקונטקסט ירד לא נקלט בשידור ("It already happened. I missed it").



אירוע `compaction_fired` ב-`log` (9K dropped, now 10K), ולצדו הסיכום המובנה שמועבר להמשך הריצה: `Data- Task, Objective`. `Already Gathered`.

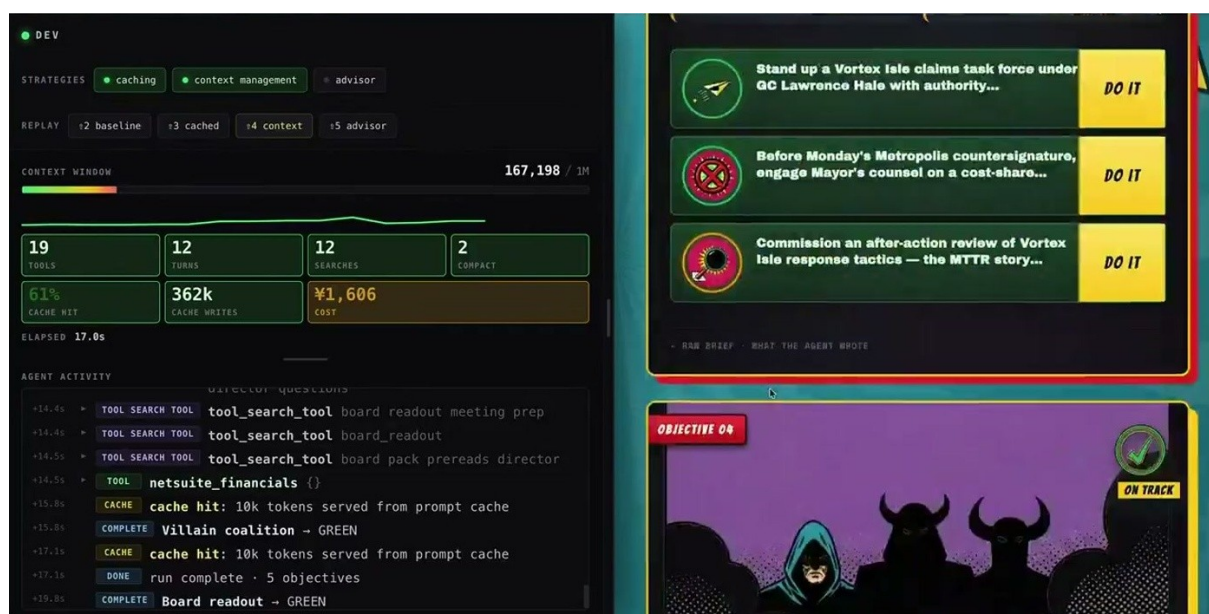
6. Advisor strategy: מודל זול מבצע, מודל יקר מייעץ

כל היעדים נטענו, אבל הדובר מעיר שבקשה אחת עולה כ-1,600 ין. הפתרון שהוא מציע לקוח מעבודה בצוות פיתוח: מהנדס ג'וניור שיכול לשאול מהנדס סניור כמה שאלות ביום משתפר מאוד, וזה עולה לסניור מעט זמן. באותו אופן נותנים ל-Sonnet או ל-Haiku כלי שמאפשר להם להתייעץ עם Opus או עם Fable, המודל שהושק באותו יום. לדבריו, Fable עולה פי 2 מ-Opus. ההיגיון: Haiku ו-Sonnet טובים בכתובת קוד וב-tool calling, ופחות טובים ב-"Bolt". deep insightful reasoning מוזכרת כלקוחה שעובדת כך.

— הדובר הראשי [20:17] You can take a small, inexpensive model and make it way better by giving it a tool to let it talk to a bigger model.

בדמו המודל המקורי הוא Opus 4.8 (כך לפי התמליל). הדובר מסביר שאפשר פשוט להחליף ל-Sonnet, אבל זה dashboard שמנכ"ל מקבל לפיו החלטות עסקיות, והוא צריך להיות גם זול וגם מדויק. כשמצב ה-Advisor מופעל, Sonnet מריץ את הכלים ופונה מדי פעם ל-Opus. במקרה ש-Rod מצא, Sonnet סימן עסקה עם Metropolis בירוק. Opus עבר על התמלולים ומצא בתחתית תמלול של Gong שראש העיר ביקש שגיבור מסוים יגיע לפתיחה, בקשה שצוות הלקוח פספס. הסטטוס של העסקה הוחלף לאדום. שם הגיבור משובש בתמליל (Cryothane או Clifane).

— הדובר הראשי [23:27] Sonnet called the advisor running Opus, and Opus actually disagreed with the decision that Sonnet had made.



הריצה במצב Advisor: ב-activity log רואים outlook_financials, tool_search_tool ו-cache hits, ולצד זה כרטיסי פעולה של העסקה עם Metropolis.

7. מה עוד הושק בפלטפורמה

בסיכום הדובר מציג שקף עם כל מה שהושק ב-2026 עד אותו רגע. השקף לא נקלט בפריימים. הוא מתעכב על שני פריטים. הראשון הוא Workload Identity Federation (WIF), שמבטל את הצורך ב-API key, ואיתו את הסיכון של מפתחות שנשמרים בקוד המקור או נשארים זרוקים. השני הוא fallback ב-Messages API. לדבריו, Fable הוא "mythos class model" (כך בתמליל) ולכן פועלים עליו safety classifiers נוספים, ושיעור החסימות שלו גבוה יותר. עם fallback אפשר לרשום מודלים חלופיים, ואם בקשה נחסמת היא עוברת אוטומטית לאחד מהם.

— הדובר הראשי [24:59] What WIF does is it removes the need for an API key.

8. מה לעשות עכשיו

- למדוד cache hit rate: לפתוח את ה-dashboard בקונסולה. אם השיעור מתחת ל-80%, זו המשימה הראשונה.

- להריץ את ה-skill של Claude API: לבקש מ-Claude Code "improve my cache hit rate", ולבדוק אילו cache control headers הוא הוסיף.
- לספור כמה טוקנים תופסות הגדרות הכלים: מעבר לכמה עשרות כלים כדאי לנסות tool search tool.
- לזהות כלים שמחזירים הרבה נתונים: מיילים, HTML ותמלולים הם מועמדים טבעיים ל-programmatic tool-calling.
- להגדיר סף ל-compaction ב-agents ארוכים: ואם כבר כתבתם מנגנון סיכום משלכם, לבדוק אם אפשר להחליף אותו, כמו שעשו ב-Hex.
- לנסות Advisor במקום להחליף מודל: Sonnet או Haiku כמבצעים ו-Opus כיועץ, ולמדוד גם עלות וגם איכות.
- לעבור ל-WIF ולהגדיר fallback: בעיקר אם אתם עובדים עם מודל שחוסם יותר בקשות.

שאלה פתוחה	למה היא חשובה
כמה עלתה הבקשה אחרי שה-Advisor הופעל?	הוא הוצג כפתרון לבעיית ה-1,606¥, אבל המספר שאחריו לא הוצג
כמה פעמים Sonnet פונה ל-Advisor, ומי מחליט מתי?	תדירות הפניות קובעת את העלות בפועל
איך tool search מוצא את הכלי ומה קורה כשהוא טועה?	החמצה של כלי עלולה לשבור את הריצה בשקט
מה הולך לאיבוד בסיכום של compaction?	ב-agents שרצים ימים, שגיאה בסיכום מצטברת
מה הקשר בין סף 400K לבין "9K dropped, now 10K" בדמו?	המספרים שהוצגו לא מתיישבים עם ההסבר
האם fallback מ-Fable למודל חלש יותר מסומן ללקוח?	בקשה שהמודל המבוקש חסם ומודל אחר ענה עליה צריכה להיות שקופה

9. מילון מונחים

מונח	משמעות
Prompt caching	שמירה בצד השרת של חלק הפרומפט שחוזר בין בקשות, כדי שלא יחושב מחדש. טוקנים מה-cache עולים 10% מהמחיר
Cache hit rate	שיעור הטוקנים בקלט שנשלפו מה-cache
KV values	הערכים המחושבים מראש שנשמרים ב-cache
Context engineering	הדיסציפלינה של ההחלטה מה נכנס לקונטקסט של המודל
Tool search tool	כלי שמחפש במלאי הכלים וטוען לקונטקסט רק את הכלי שנדרש
Programmatic tool calling	המודל כותב קוד שמעבד את תוצאת הכלי ומכניס לקונטקסט רק את מה שצריך
Compaction	סיכום אוטומטי של ההיסטוריה כשהקונטקסט מגיע לסף, ואחריו המשך הריצה
Advisor strategy	מודל זול שמבצע את העבודה ומתייעץ עם מודל חזק ויקר יותר
Workload Identity Federation (WIF)	הזדהות מול ה-API בלי API key קבוע
Fallback	רשימת מודלים חלופיים שבקשה עוברת אליהם אוטומטית אם היא נחסמת
Claude API skill	skill שמותקן ב-Claude Code ומכיר את הדרך הנכונה לעבוד עם ה-API, כולל caching