בונים את Claude API לחזית היכולות

שני מהנדסים מצוות Claude Platform מדברים על knobs ב-Messages API, מיגרציות בין מודלים, עלויות ו-safeguards

Jake Dejno, Lucca Siaudzionis · Claude · 12.11.2026

משך: כ-18 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו שיחה של כ-18 דקות מערוץ Claude. אין בה מצגת, רק שני אנשים ליד שולחן. Jake Dejno (בכתובית: Member of Technical Staff, Head of API Engineering) מנחה את השיחה, ו-Lucca Siaudzionis (בכתובית: Member of Technical Staff, Platform Intelligence Lead) עונה על רוב השאלות. בחלק מהנושאים התפקידים מתחלפים, ו-Jake עונה בעצמו, למשל על גרסאות ועל intelligence per token. הנושאים: איך עובד ה-Messages API, איך מחליטים אילו פיצ'רים לשחרר, איך מתנהלות מיגרציות בין מודלים, איך שולטים בעלויות, איך מאזנים בין גישה לבטיחות במודלים Mythos ו-Fable, ומה הכיוון הבא של ה-API.

הטענה המרכזית: הערך של ה-API לא נגמר במודל עצמו. הוא נמצא גם ב-knobs, כמו effort, thinking ו-prompt caching, שנותנים שליטה שאי אפשר להשיג ב-prompting בלבד. לפי הדוברים, ה-knobs האלה נבנים בלולאה הדוקה עם המחקר ונמדדים ב-evals. המסר החוזר למפתחים: להריץ evals משלכם, למקסם את ה-prompt caching, ולבחון מחדש את ה-harness בכל פעם שמגיע מודל חדש.



שני הדוברים ליד שולחן העבודה באולפן. משמאל Jake Dejno, מימין Lucca Siaudzionis (הזיהוי לפי כתוביות השם שמופיעות בתחילת הסרטון).

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות בעמוד אחד, ומה לא נאמר בשיחה.
- **2. Messages API ו-knobs:** מה ה-API נותן מעבר ל-prompting ואיך מחליטים מה לשחרר.
- **3. גרסאות ומיגרציות:** מחרוזת המודל כמנגנון גרסאות, המעבר ל-adaptive thinking וה-harness שמזדקן.
- **4. עלויות:** prompt caching, יעד ה-90% ומעבר מ-cost per token ל-intelligence per token.
- **5. Mythos ו-Fable:** אותם model weights עם safeguards שונים, distillation ו-evals.
- **6. מה הלאה:** פיצ'רים שמחכים למודל הבא, ו-Claude Managed Agents מעל ה-Messages API.
- **7–8. פעולות ומילון:** מה כדאי לעשות עכשיו, שאלות פתוחות ומילון מונחים.

הערת מקור המסמך הופק אוטומטית מהתמלול ומהפריימים של ההקלטה. לסרטון אין שקפים, ולכן מופיע רק פריים אחד שמזהה את הדוברים. ייתכן ששמות ומונחים שובשו בתמלול האוטומטי (ASR). למשל, Claude מופיע בתמלול כ-Cloud, ו-distillation מופיע פעם אחת כ-desolation. הציטוטים מועתקים מילה במילה מהתמלול, כולל השיבושים, ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

השיחה היא בעצם מדריך מוצר קצר שמוגש כדיאלוג פנימי. אין בה הכרזות או מספרים קשים, חוץ מיעד ה-90% ל-prompt caching. יש בה סדרת עקרונות שמסבירים איך צוות ה-API חושב, ומה הצוות מצפה שמפתחים יעשו בכל פעם שמגיע מודל חדש.

- **Knobs נותנים יותר מ-prompting:** לדברי Lucca, effort ו-knobs דומים פותחים רמת התאמה שאי אפשר להשיג בניסוח prompt. אותו מודל ב-high effort וב-low effort מתנהג כמעט כמו שני מודלים שונים.
- **שינוי effort או system prompt באמצע שיחה בלי לשבור את ה-cache:** בעבר כל שינוי כזה שבר את ה-prompt caching, ולמשתמשים זה היה "deal breaker". היום אפשר להוריד effort לבקשה קלה בתוך flow ארוך בלי לאבד את החיסכון (בבטא, לפי תיאור הסרטון).
- **פיצ'רים נולדים בלולאה דו-כיוונית עם המחקר:** לפעמים יכולת שהחוקרים מזהים הופכת לכלי. כך נולד computer use. לפעמים בקשה של לקוחות מתורגמת ליעד אימון למודלים הבאים.
- **מחרוזת המודל משמשת כמנגנון גרסאות:** לפי Jake, הערך של מודל חדש גבוה מספיק כדי שמשתמשים יסכימו לעשות התאמות, ולכן אפשר להוציא שינויים מהר.
- **Deprecation מתרחש כשהמודל כבר לא מאומן על הדרך הישנה:** דוגמה: המעבר מ-extended thinking ל-adaptive thinking. נשארו toggle ו-beta headers לחזרה אחורה, אבל בסוף אין טעם להשאיר knob שהמודל כבר לא מכבד.
- **Prompt caching הוא ההמלצה הראשונה לכל לקוח:** היעד הוא hit rate של עד 90%, ובצוותי המכירות זו השאלה הראשונה שמעלים מול לקוח.
- **למדוד intelligence per token ולא רק cost per token:** מודל חכם ויקר יותר עשוי לסיים את המשימה בהרבה פחות טוקנים. לפי Jake, לקוחות שנשארים ב-Haiku או Sonnet מפספסים את זה לפעמים.
- **Mythos ו-Fable הם אותם weights עם classifiers שונים:** הגישה ל-Mythos עוברת דרך תהליך KYC, ו-Fable פתוח לכולם. נוסף לזה הוכנסו thinking protections נגד distillation.
- **ה-API נבנה בעיקר עבור agents:** Claude Managed Agents בנוי מעל ה-Messages API, וכל לקח שהצוות לומד בבניית agents עובר לכל המשתמשים.

מה לא נאמר

- **אין פרטי API:** לא הוצג איך משנים effort באמצע שיחה בפועל (פרמטר, header) ולא אילו מגבלות יש לבטא.
- **אין מספרים:** מלבד יעד ה-90% ב-caching לא נמסרו מחירים, נתוני חיסכון, תוצאות evals או לוחות זמנים ל-deprecation.
- **Mythos מול Fable נשאר מעורפל:** לא פורט מה ה-classifiers חוסמים ב-Fable, ולא תהליך ה-KYC עצמו. השם Cyber Verification Program מופיע רק בתיאור הסרטון.
- **Thinking protections:** הוזכר שיצאה סדרת שינויים נגד distillation, אבל לא הוסבר מה השתנה בפועל למפתחים.
- **הטענה ש-knobs עדיפים על prompting** נשענת על evals פנימיים שלא הוצגו.

2. ה-Messages API, ה-knobs וההחלטה מה לשחרר

Lucca מתאר את ה-Messages API כאבן הפינה של הפלטפורמה. המבנה מוכר: system, תורות של user וקריאות לכלים באמצע. מעל המבנה הזה יש שכבה של knobs שמשנים את ההתנהגות של Claude. ברירות המחדל משקפות את הדעה של הצוות, שיושב קרוב מאוד למחקר, על מה שגורם ל-Claude לעבוד הכי טוב. לכן העתקה של הבקשה הבסיסית מהתיעוד אמורה לתת את חוויית ה-out-of-the-box המומלצת. ה-knobs מוסיפים שליטה שאי אפשר להשיג רק בניסוח ה-prompt.

> **Lucca Siaudzionis** — But with those knobs, you can do more than any prompting outside could ever do.

ההחלטה מה לשחרר מתקבלת בשיחה דו-כיוונית. בכיוון אחד, החוקרים מזהים יכולת חדשה, כמו השיפור בשימוש במחשב שהפך בסוף ל-computer use tools. בכיוון השני, בקשות של לקוחות מתורגמות ליעדי אימון למודלים

הבאים. מבחינת Lucca, הדוגמה הכי חזקה לערך של knob היא היכולת לשנות effort ו-system prompt באמצע שיחה. גם קודם אפשר היה לשנות אותם, אבל השינוי שבר את ה-cache. כל פיצ'ר כזה עובר evals לפני שהוא יוצא, והצוות ממליץ שכל לקוח יריץ גם evals משלו, כי ה-API גנרי וה-use case של כל לקוח הוא ייחודי.

> **Lucca Siaudzionis** — And sometimes a user has an agentic flow that has been operating under high effort, and they know that the next request is actually easier. But if they change the effort to low, the cache is broken, and that defeats the whole purpose of their saving cost.

3. גרסאות, מיגרציות ו-deprecation

Jake מגיע מתפקידים שבהם ניהול גרסאות של API היה חשוב וגם כואב. לדבריו, מודלים מייצרים מנגנון גרסאות חדש: מחרוזת המודל, גרסת ה-API, יציבות ה-prompt ומתגים שמאפשרים להכניס שינויים בהדרגה. בגלל שהקפיצה ביכולות בין מודלים גדולה, משתמשים מוכנים לעשות התאמות, וזה מאפשר לצוות להוציא שינויים מהר. ובכל זאת, את ה-harness צריך לבדוק ב-evals בכל מעבר: כמות טוקנים, איכות ותוצאת המשימה.

> **Jake Dejno** — Models provide such an interesting version mechanism for us, right?

Lucca מבחין בין שני סוגים של versioning מרומז. שינוי שמוסיף יכולת לא מעורר תלונות. שינוי שמסיר יכולת שמישהו נשען עליה הוא הקשה. הדוגמה שלו היא המעבר מ-extended thinking ל-adaptive thinking. ה-adaptive כבר ניצח ב-evals, ולכן הפך לברירת המחדל, אבל לתקופה מסוימת נשאר toggle לחזרה לשיטה הישנה. בנוסף יש beta headers שקונים זמן לעדכן את ה-harness. הסוף נקבע כש-"at some point models stop being trained for this old way". מבחינת הצוות, knob שלא עושה את מה שהוא אמור לעשות מזיק יותר מאשר להסיר אותו.

> **Lucca Siaudzionis** — But at some point, if a model is 10 times smarter than the previous model was, then the harness can start to be a limitation.

4. עלות: prompt caching ו-intelligence per token

לפי Lucca, "a lot of roads lead to prompt caching". ב-turn של agent, כל בקשה דומה לקודמת ורק מוסיפה עליה, ולכן הבלוקים שכבר עובדו נשמרים ב-cache. הוא מציין שזו ההמלצה הראשונה לכל לקוח שעוד לא עשה אופטימיזציה ל-caching, ושצוותי המכירות שואלים עליה בכל שיחה עם לקוח.

> **Lucca Siaudzionis** — and you get it to up to 90% from caching, because those are the dream numbers to achieve, and it really translates to cost, and it's a win-win all around.

הנושא השני הוא cost per task מול cost per token. חלק מהפיצ'רים לא משנים את המחיר לטוקן, אבל משנים את מספר הטוקנים שהמשימה צורכת. Jake מוסיף מימד: "the cost per token also needs the dimension of intelligence per token", כלומר כמה מהר המודל מגיע לתוצאה. מודל ברמת Fable עשוי לצרוך הרבה פחות טוקנים מ-Haiku עבור אותה תוצאה. Jake רואה לקוחות שנשארים ב-Haiku או Sonnet בלי לבדוק אם מודל חכם יותר היה זול יותר לכל משימה. גם הכיוון ההפוך נכון: משימה שבעבר רק המודל החזק ביותר פתר עשויה לעבור למודל זול יותר בדור הבא.

> **שמות לא ודאיים** — בתמלול מופיעה הדוגמה "Fable 5" מול "Opus 5". Fable מוזכר בתיאור הסרטון, אבל ייתכן שהתמלול שיבש את מספרי הגרסה ואת שמות המודלים. כדאי להתייחס לזה כדוגמה להמחשה ולא כמיפוי רשמי של מודלים.

5. Mythos ו-Fable: אותו מודל עם safeguards שונים

Lucca מתאר שני דחפים שמופיעים עם כל מודל חדש: להעביר אותו מהר ללקוחות, שיש להם יותר רעיונות מלצוות, ובהם גם חוקרים במדע, ולצד זה לשאול מה גורם זדוני יעשה איתו. כל מודל ב-API רץ עם classifiers נגד שימוש לרעה. הפיצול בין Mythos ל-Fable נולד מההכרה שיש הרבה עבודה מועילה לעשות בתחום הסייבר. מי שעובר תהליך KYC מקבל גישה ל-Mythos, ומי שלא עובר אותו משתמש ב-Fable. לפי תיאור הסרטון, הגישה ל-Mythos עוברת דרך Cyber Verification Program.

> **Lucca Siaudzionis** — And it's the same intelligence, the same model weights at the end of the day. And it's just about how much you can do with this model based on classifiers.

הצד השני של הבטיחות הוא distillation. מודל שזוקק ממודל חזק יהיה חלש יותר, אבל עדיין חזק, ו-"That model being unsafeguarded in the wild is an incredibly dangerous position for the world". לכן לאחרונה יצאה סדרת שינויים ב-thinking protections, ולדבריו היא מתועדת כאמצעי נגד distillation. האיזון מול חוויית המפתח מושג בעזרת evals: לכל פיצ'ר ציבורי קדמו בערך עשר וריאציות פנימיות. הם גם נתקלים ב-eval saturation, כלומר eval שכל המודלים כבר פותרים ולכן הוא מפסיק להיות משמעותי.

6. מה הלאה: פיצ'רים שמחכים למודל ו-agents

לפי Lucca, הדבר המרגש בכל מודל חדש הוא יכולות חדשות. הצוות מריץ מחדש את כל הפיצ'רים הקיימים, וגם פיצ'רים שלא שוחררו כי לא עמדו ברף. כשאחד מהם עובר את הרף, הוא יוצא. אותו עיקרון מומלץ גם למפתחים: לנסות שוב את מה שכמעט עבד. Lucca מוסיף שרוב ה-API מתוכנן עבור agents. יש Claude Managed Agents (בבטא, לפי התיאור), כלים בצד הלקוח כמו bash ו-text editor שעברו evals, ושליטה ב-thinking וב-effort. Jake מזכיר שגם Claude Code ו-Claude.ai בנויים על אותו API.

> **Lucca Siaudzionis** — Everything else next to that matters, but nothing matters as much as model intelligence.

Lucca מסיים בנקודה ארכיטקטונית: "Cloud Management Agents is built on top of the Messages API". כשצוות ה-API מבין מה נדרש לבניית agents, הוא מכניס את היכולת ל-Messages API עצמו. כך גם מי שבונה agent ישירות על Messages API מקבל את אותם לקחים.

7. מה לעשות עכשיו

- **למדוד את ה-cache hit rate:** להציב יעד שמתקרב ל-90%, ולבנות את הבקשות כך שכל בקשה רק מוסיפה על הקודמת.
- **להוריד effort לבקשות קלות בתוך flow:** לבדוק את האפשרות לשנות effort באמצע שיחה (בבטא) במקום לפתוח שיחה חדשה ולשבור את ה-cache.
- **לבנות evals משלכם:** לפני כל החלפת מודל או knob, למדוד טוקנים, איכות ותוצאת משימה מול ה-use case שלכם.
- **להשוות cost per task בין מודלים:** להריץ את אותה משימה על מודל חזק יותר ועל מודל זול יותר ולהשוות את העלות הכוללת, לא את המחיר לטוקן.
- **לנקות את ה-harness בכל מעבר מודל:** לחפש רכיבים שפיצו על חולשות של המודל הקודם ועכשיו רק מגבילים.
- **לנסות שוב את מה שנכשל:** prototypes שלא עבדו עם המודל הקודם שווים הרצה חוזרת.

שאלה פתוחה	למה זה חשוב
איך בדיוק משנים effort באמצע שיחה, ומה מגבלות הבטא?	זה הפיצ'ר שהוצג כהכי משמעותי לחיסכון, ולא הוצג לו שום פרט טכני.
מה ה-thinking protections שינו בפועל בתשובות ה-API?	שינויים נגד distillation עלולים להשפיע על מה שמפתחים רואים מה-thinking.
אילו בקשות Fable חוסם ו-Mythos מאפשר?	מפתחי אבטחה צריכים לדעת אם כדאי להם לעבור תהליך אימות.
כמה זמן יישאר ה-toggle ל-extended thinking?	נאמר רק שמאריכים את התקופה ככל האפשר, בלי תאריך.
האם יהיה כלי מובנה להשוואת עלות לכל משימה בין מודלים?	Jake אמר שהוא רוצה לבנות מוצר כזה, אבל לא התחייב.

8. מילון מונחים

מונח	משמעות
Messages API	ה-API הבסיסי של Claude, שבנוי מתורות system, user ו-tool calls
effort	פרמטר שקובע כמה מאמץ חשיבה המודל משקיע

מונח	משמעות
prompt caching	שמירה של בלוקים שכבר עובדו, כדי שבקשות חוזרות יהיו זולות ומהירות יותר
adaptive thinking	מצב חשיבה שהחליף את extended thinking כברירת מחדל
beta header	כותרת בבקשה שמפעילה או מבטלת התנהגות בבטא
harness	הקוד והתשתית שעוטפים את המודל באפליקציה
evals	מבחנים שמודדים איכות, עלות ותוצאה
eval saturation	מצב שבו כל המודלים פותרים eval, ולכן הוא כבר לא מבחין ביניהם
intelligence per token	כמה התקדמות לתוצאה המודל מייצר לכל טוקן
classifiers	מודלי סינון שמזהים שימוש לרעה
KYC	אימות זהות לקוח לפני מתן גישה מורחבת
distillation	אימון מודל חדש על הפלטים של מודל חזק יותר
Claude Managed Agents	שירות agents מנוהל שבנוי מעל ה-Messages API