

האם ה-Agent Harness מת?

מודלים חכמים יותר ספגו את מה שה-harness הוסיף ל-coding. נשאלו לו שלושה תפקידים: לפתוח את החזית הבאה, לספק תשתית ואמון, ולהוריד עלויות.

Jediah Katz · AI Engineer · 19.10.2026

משך: כ-18 דקות · YouTube

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הרצאת breakout מכנס AI Engineer New York (14 באוקטובר 2026, Shannon Room). מרצה בה Jediah Katz, שהצטרף ל-Cursor בתחילת 2025. לדבריו הוא המהנדס הוותיק ביותר בצוות ה-agent harness שם, ועבר ל-SpaceX במסגרת רכישת Cursor. ההרצאה בנויה סביב שאלה אחת: האם ה-harness, השכבה שעוטפת את המודל ומחברת אותו לעולם, הפך למיותר עכשיו כשהמודלים חכמים יותר?

התשובה שלו כפולה. החלקים שהוסיפו ל-harness "אינטליגנציית קוד" באמת מתים: מחקרים מראים שבחירת ה-harness כמעט לא משנה את הציון ב-benchmarks, והמודלים למדו לבד את מה שה-harness הוסיף להם באופן מלאכותי. מצד שני, ה-harness חשוב יותר מתמיד משתי סיבות. הוא זה שפותח למודל חזיתות חדשות (הדוגמה המרכזית היא Grok Bot), והוא מחזיק דברים שהמודל לא יכול לספוג: תשתית ריצה, ארון ואבטחה, יעילות בעלות.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר.
- 2. **מה יש בתוך harness:** האבולוציה של התפקיד וחלוקה לשתי שכבות, runtime ו-context/orchestration.
- 3. **שיא ה-harness והתגובה הנגדית:** הדפדפן שנבנה מאפס, הטענה ש"ה-harness מת" והמחקרים שתומכים בה.
- 4. **ה-data flywheel ו-Grok Bot:** למה ה-harness מייתר את עצמו, ואיך הוא פותח את הסבב הבא.
- 5. **מה שהמודל לא יכול לספוג:** תשתית, ארון ואבטחה, יעילות.
- 6. **מה הלאה:** shims, primitives, שכבת הפשטה גבוהה יותר, ותחזית ה-90%.
- 7-8. **מסקנות ומילון:** פעולות מומלצות, שאלות פתוחות ומונחים.

הערות מקור המסמך הופק אוטומטית מהתמלול ומהשקפים של ההקלטה. התמלול האוטומטי משבש שמות ומונחים: שם המרצה תומלל "Jadaya", ושמות המוצרים תומללו בגרסאות כמו "GrokBot" ו-"Crock-Bot" או "Cloud Code" במקום Claude Code. במקומות שבהם אין ודאות זה מצוין. הציטוטים מועתקים מילה במילה ונבדקו מול חותמת הזמן שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

ההרצאה טוענת שהשאלה "האם ה-harness מת" מבלבלת בין שני דברים שונים. מצד אחד יש טלאים שתיקנו חולשות של המודל בכתיבת קוד, וכאן המודלים אכן ניצחו. מצד שני יש תשתית, בקרה ויעילות, ושם ה-harness נשאר הכרחי. המרצה מבסס את הטענה על שני מחקרים, על הניסיון ב-Cursor ועל Grok Bot, מוצר שבו הצוות דווקא מחזיר הנחיות ו-guardrails שהוסרו ממנו בעבר.

- **ה-harness כבר לא מוסיף אינטליגנציית קוד:** מחקר מ-UC Berkeley מצא שבמודלים החדשים בחירת ה-harness (Pi, Codex, Claude Code) כמעט לא הזיזה את הציון ב-SWE-Bench Lite. מחקר של Apple ו-EPFL הגיע למסקנה דומה ב-MLBench.
- **ב-Cursor הסירו guardrails:** למשל מודל קטן שהיה אחראי רק להחלת עריכות קוד, והנחיות ספציפיות לקוד על חקירת ה-codebase ובדיקת העבודה.
- **ה-data flywheel:** ה-harness מקרב משימות קשות, יותר משתמשים מבצעים אותן, יותר דאטה מגיע למעבדות, והמודל לומד את המיומנות בעצמו. בסוף אפשר למחוק את הטלאי מה-harness.
- **ה-harness פותח את החזית הבאה:** ב-Grok Bot מחזירים הנחיות ו-guardrails, כי בתחומים החדשים (מצגות, הזמנת נסיעות, תקשורת, ההחלטה אם לפעול או לשאול) המודלים עדיין לא שולטים.
- **תשתית:** agents שעובדים שעות וימים לא יכולים לרוץ על לפטופ שנכנס למצב שינה. הם חייבים לרוץ בענן.
- **אמון ואבטחה:** הרשאות מצומצמות לפי agent ולפי אדם, ו-guardrails נגד פעולה קטסטרופלית או חשבון API של 10,000 דולר.
- **יעילות:** שני המחקרים מצאו הבדל משמעותי בעלות לכל משימה. ב-Grok Bot הורידו עלות והאצו את המוצר דרך cache hit rate, יעילות טוקנים ותמהיל מודלים, לדבריו בלי פגיעה באיכות.
- **shims מנהלים מרכזית:** כל טלאי דטרמיניסטי נרשם בלוג בכל פעם שהוא מופעל. bot עוקב אחרי הדשבורד ופותח PR להסרה כשטלאי מפסיק להיות מופעל.
- **תחזית:** בתוך שישה חודשים 90% מעבודת ה-agents תרוץ מחוץ ללפטופ.

מה לא נאמר

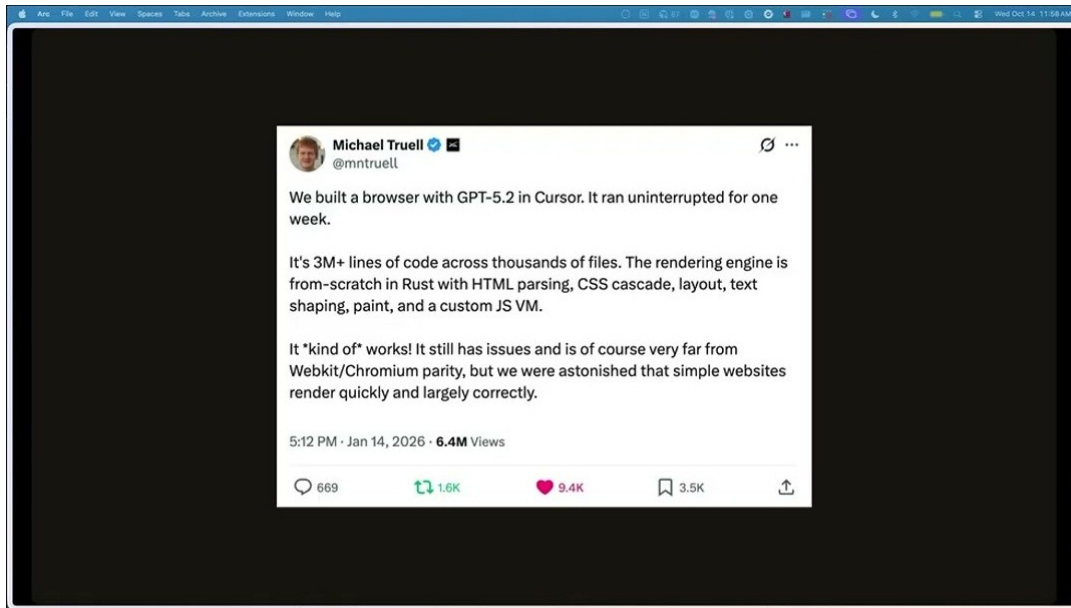
- **אין מספרים על החיסכון:** הטענה על הורדת עלות וזמן תגובה ב-Grok Bot נשארת איכותית. בשקף מופיעים -65% ו-72%, אבל לא מוסבר מה הם מודדים.
- **אין פירוט של המחקרים:** לא הוצגו ציונים, גדלי מדגם או אילו מודלים נבדקו. נאמר רק "שבעה מודלי frontier".
- **מדד האיכות לא הוגדר:** "בלי פגיעה באיכות" ב-Grok Bot נאמר בלי לציין איך זה נמדד.
- **לתחזית ה-90% אין בסיס:** זו הערכה אישית, ולא הוצג בסיס כמותי.
- **אין שום פרט על ארכיטקטורת ה-coordinator/worker:** לא נאמר כמה workers יש, איך מתקשרים ביניהם או איך מטפלים בכשלים. המרצה אמר במפורש שזה חומר להרצאה אחרת.

2. מ-Prompt Engineer ל-harness: מה יש בפנים

המרצה מתאר שלושה שלבים. ב-2023 התואר החם היה Prompt Engineer. באמצע 2025 עבר המרכז ל-Context Engineer, מי שמרכיב בזמן אמת את ההקשר הטוב ביותר בשביל המודל. רק בתחילת 2026 התעשייה התיישבה על המונחים agent harness ו-harness engineering. בלשונו, ה-harness הוא "העיניים והידיים" של המודל: המשתמשים מתחברים אליו דרך אפליקציות, והוא מחבר את המודל למחשבים ולשירותים חיצוניים כמו MCP.

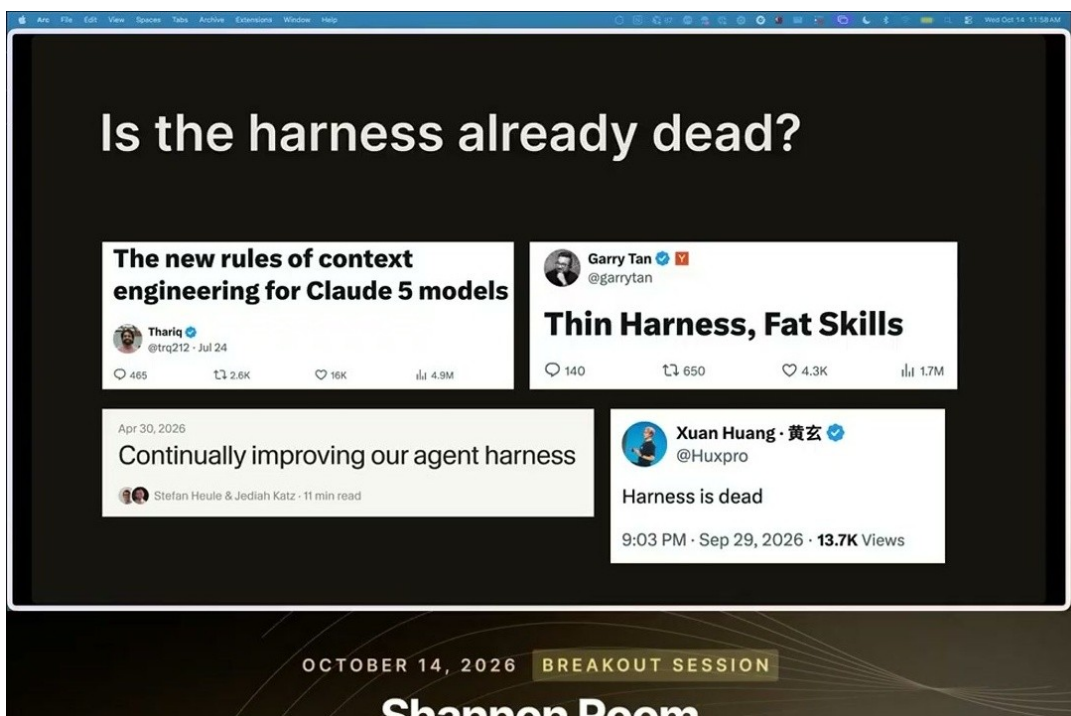
הוא מחלק את ה-harness לשתי שכבות. ה-runtime הוא היסוד: הוא קובע איפה רצה לולאת ה-agent ואיך נשמר state כדי שאפשר יהיה להמשיך עבודה. זו לדבריו הנדסת תוכנה קלאסית. שכבת ה-context וה-orchestration היא "המערב הפרוע": היא קובעת איזה מודל נקרא ומתי ומה הוא רואה, כלומר prompts, tools, memory, compaction. "הרבה טריקים שאתם אולי לא מכירים". בשכבה הזו עובד המרצה עצמו.

3. שיא ה-harness והתגובה הנגדית



ציור של Michael Truell מ-14 בינואר: דפדפן שבנה עם GPT-5.2 ב-Cursor, רץ שבוע ברציפות, יותר מ-3 מיליון שורות קוד באלפי קבצים, rendering engine שנכתב מאפס ב-Rust.

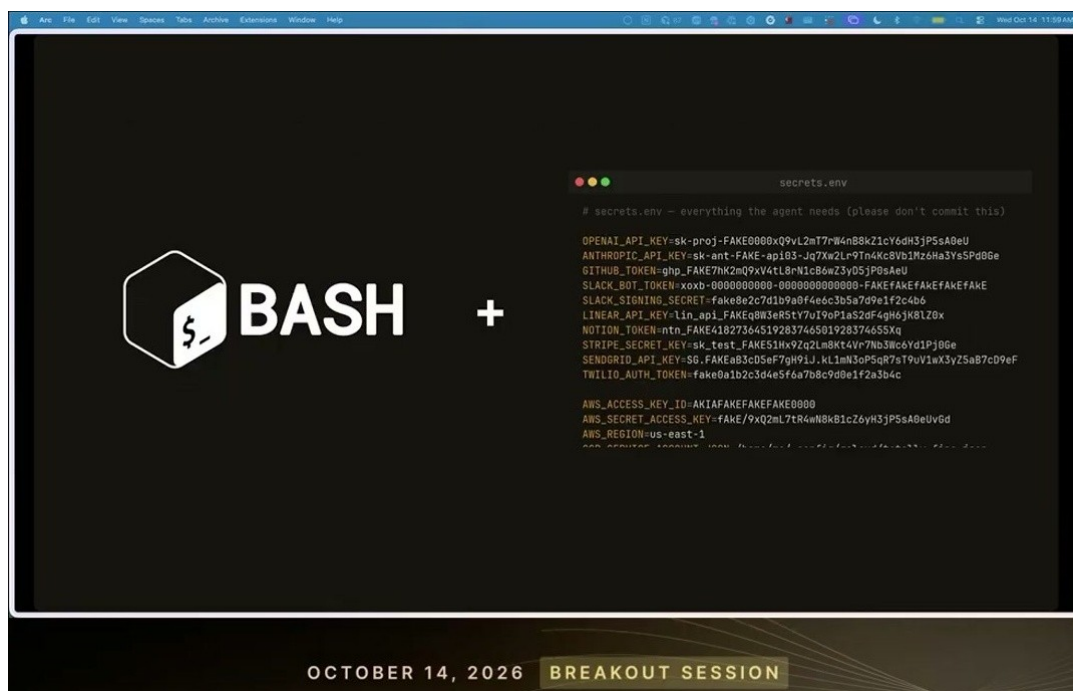
בתחילת השנה, לדבריו, הגיעו ל"Cursor harness": peak harness הודיעה שבנתה דפדפן מאפס בעזרת harness חדש וארוך-טווח (השם תומלל "Grindswarm" והאיות לא ודאי). זה היה harness מרובה-מודלים שתזמר הרבה agents על הרבה מכונות, והריצה שלו הייתה יקרה מאוד: "It actually took down Cursor maybe once or twice to get this demo".



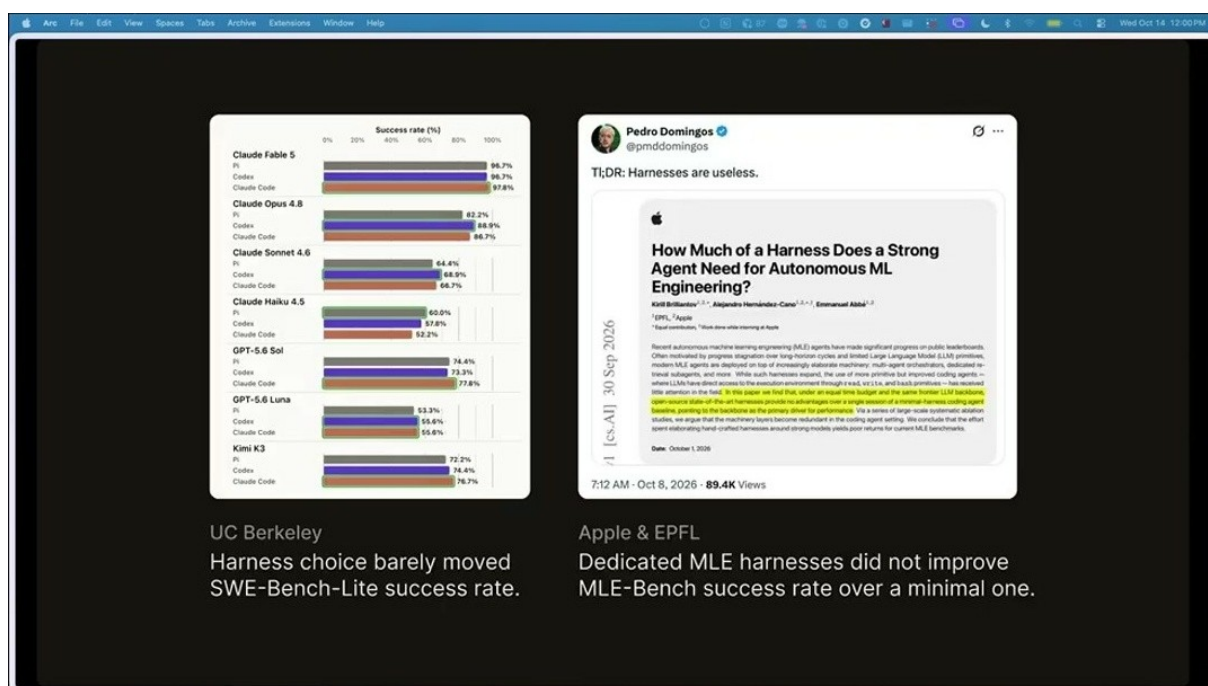
השקף "Is the harness already dead?": ציורים על "The new rules of context engineering for Claude 5 models", "Thin Harness", הפוסט של Garry Tan "Fat Skills", וציור "Harness is dead".

התגובה הנגדית הגיעה תוך חודשים ספורים. איש מ-Anthropic (תומלל "Tariq", כנראה Thariq) סיפר שהם מחקו חלק גדול מה-system prompt של Garry Tan. Claude Code מ-YC טען שה-harness ייעלם ורק skills יישארו חשובים. גם המרצה ועמית בשם Stefan (השם לפי התמלול) כתבו על הסרת Cursor מ-guardrails שנועדו לכסות

על פערים במודל. דוגמאות: מודל קטן שתפקידו היחיד היה להחיל עריכות קוד, כי מודלי ה-frontier לא עשו את זה טוב, והנחיות על חקירת ה-codebase ובדיקת העבודה.

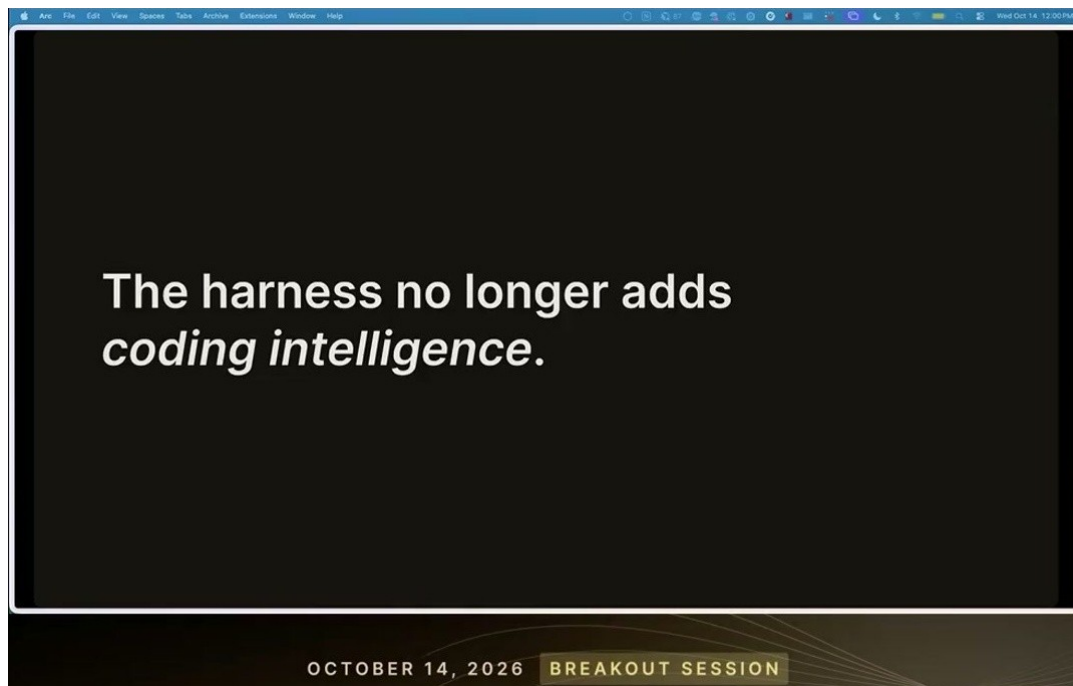


ניסוי מחשבת: harness מינימלי שכולל רק BASH, ולצידו קובץ עם כל הסודות וה-API keys של המשתמש.



שני המחקרים: Apple & EPFL ("Dedicated MLE harnesses did not improve MLE-Bench success rate over a minimal one") ו-UC Berkeley ("Harness choice barely moved SWE-Bench-Lite success rate").

המרצה שאל את הקהל אם harness שיש בו רק shell וכל הסודות של המשתמש יצליח ב-coding evals כמו harness מתוחכם. רוב הקהל ענה שלא. המחקרים אומרים אחרת: ב-UC Berkeley בדקו שבעה מודלי frontier על כמה harnesses, ובמודלים החדשים בחירת ה-harness לא שינתה באופן משמעותי את הציון ב-SWE-Bench-Lite. מחקר של Apple ו-EPFL מצא ש-harness ייעודי ל-MLE לא שיפר את התוצאה ב-MLE-Bench.



שקף המסקנה: "The harness no longer adds coding intelligence"

Jediah Katz A fancy harness in this day and age no longer adds coding intelligence. I think that is —
.true

4. ה-data flywheel וגrok bot

ההסבר שלו לתופעה הוא ה-data flywheel. ה-harness מקרב משימות קוד קשות יותר, יותר משתמשים מבצעים אותן בהצלחה עם agents, יותר דאטה משותף עם המעבדות, ובסוף המודל לומד את המיומנות בעצמו. בשלב הזה אפשר למחוק מה-harness את הטלאי שאפשר את המשימה מלכתחילה. בלשונו: "the harness is actually making itself obsolete through what I call the data flywheel". מיומנויות כמו חקירת ה-codebase, עבודה עקשנית לאורך זמן (למשל goal) והרצת בדיקות כבר נלמדו במודל עצמו.

כאן נכנס Grok Bot, שהמרצה מתאר כ-"our autonomous colleague who gets work done for you, not just coding, but all kinds of knowledge work". ה-harness שלו לא מבוסס על שיחות חד-פעמיות. יש בו coordinator bots ארוכי-חיים שמתזמרים צי של worker agents. האנלוגיה שלו היא עורך דין בכיר שמחזיק את האסטרטגיה ומחלק את העבודה לצוות של paralegals. ההפך ממה שקרה ב-Cursor: "in GrokBot, we've actually found ourselves putting back a lot of these things, adding new prompting, adding new guardrails". התחומים שהוא מונה: flows מרכזיים כמו בניית מצגות והזמנת נסיעות, תקשורת שמעכנת את המשתמש בלי להציף אותו, ההחלטה אם לפעול או לשאול (למשל לפני שליחת מייל לעמיתים), ולמידה של דרך העבודה של המשתמש דרך memories.

הרעיון המרכזי Grok Bot סוגר את הלולאה ב-flywheel. כשהמודל הופך יכולת כמו coding לסחורה, ה-harness פותח חזית חדשה וה-flywheel מתחיל סבב נוסף. החזית יכולה להיות מחוץ לתוכנה, או בחלקים של עבודת המהנדס תוכנה שאינם כתיבת קוד.

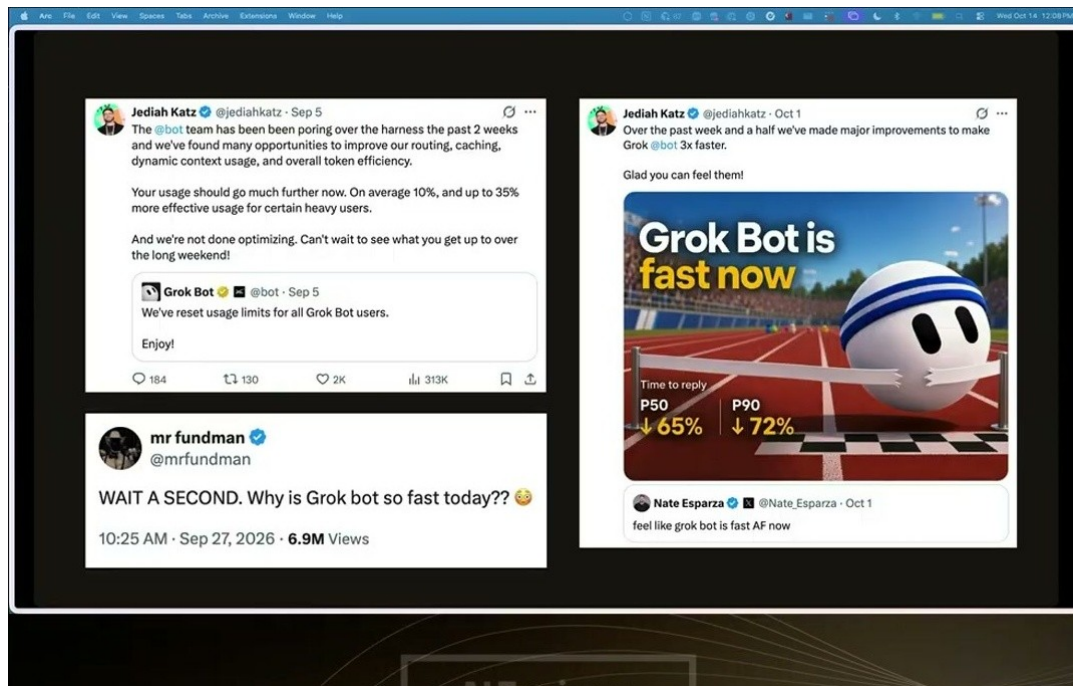
דוגמה נוספת היא ה-software factory: כמה agents שמכסים יחד את כל מחזור פיתוח התוכנה. המרצה מתאר איך זה עובד אצלם. באג מדווח בערוץ, אוטומציה ממיינת אותו מיד, cloud agent מנסה לתקן, ה-agent מצרף סרטוני לפני ואחרי כהוכחה לתיקון, ואחרי ה-merge מוצר Rollouts מפעיל agent נוסף שעוקב אחרי בעיות ב-production. "So you can think of all this as almost like a meta harness, which is orchestrating many agents together".

5. מה שהמודל לא יכול לספוג: תשתית, ארון, יעילות

תשתית: agents עברו מעבודה של דקות על משימה טיפוסית לשעות, לימים ומעבר לכך. "And there's just no way that this can work on one person's laptop, which might go to sleep להריץ אותו בענן.

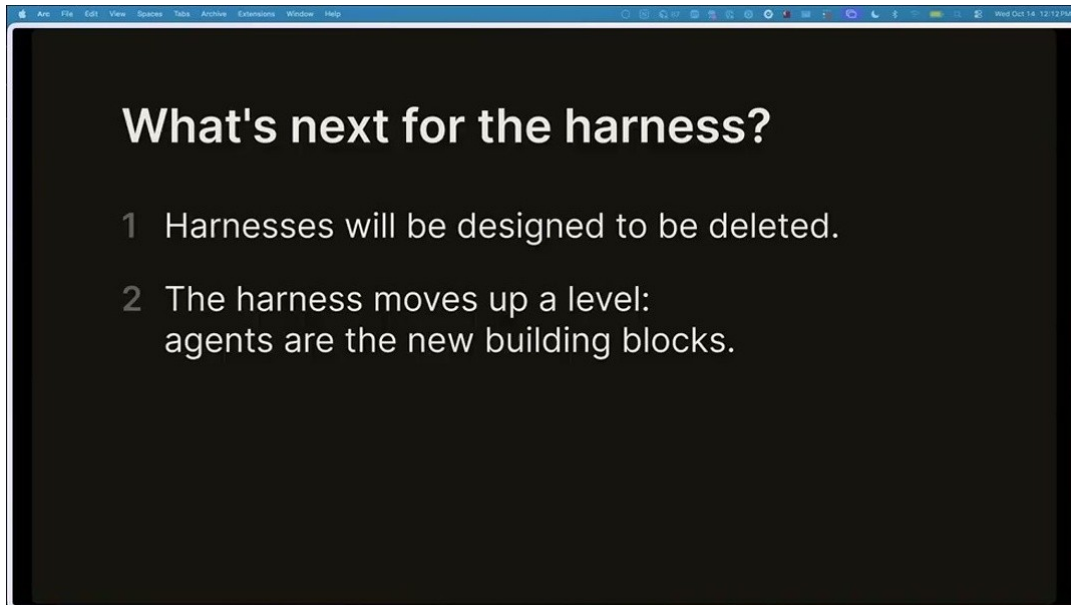
אמון ואבטחה: נכון ש-harness עם bash וכל המפתחות בקובץ פותר משימות קוד, אבל ראש האבטחה "יגיד משהו" על זה. ארגונים צריכים הרשאות שמוגבלות ל-agents ספציפיים ולאנשים ספציפיים, וכל אחד צריך guardrails כדי לא להתעורר לפעולה קטסטרופלית או לחשבון API של 10,000 דולר.

יעילות: אותם שני מחקרים מצאו ש-"both of those studies actually found that there was a significant difference in cost per task". בחודש האחרון המרצה עבד על אופטימיזציה של harness של Grok Bot. השיפורים ב-cache hit rate, ביעילות הטוקנים ובתמהיל המודלים הורידו משמעותית את העלות ואת זמן התגובה, ולדבריו בלי פגיעה באיכות. ככל שעבודה רבה יותר עוברת ל-agents, ל-harness יעיל יש יותר ערך.



תגובות ברשת לשיפור המהירות: "Grok Bot is fast now" עם המספרים -65% ו-72% (לא מוסבר מה הם מודדים), וציוץ "WAIT A SECOND. Why is Grok bot so fast today?".

6. מה הלאה ל-harness



"What's next for the harness?": (1) harnesses יתוכננו כך שיהיה קל למחוק אותם, (2) ה-harness עולה רמה ו-agents הופכים ליחידת הבנייה הבסיסית.

לפי המרצה ה-flywheel מאיץ, ו-"we've entered a new age where the product is once again moving faster than the model". המשמעות היא שה-harness יסתבך עוד לפני שינוקה, ולכן צריך לתכנן אותו כך שיהיה קל להוסיף פיצ'רים ולהסיר אותם. ביטוי אחד לכך הוא שפחות מהמנגנונים הפנימיים של ה-harness נחשפים למשתמש. הוא מצטט את Hyrum's Law (תומלל "Hiram's Law"): כל מה שחושפים למשתמשים, הם יתחילו להסתמך עליו. פחות שקיפות פירושה פחות ניהול שינויים ואיטרציה מהירה יותר.

ביטוי שני הוא primitives: ממשקי בסיס שאפשר להרכיב ולהרחיב. הדוגמה שלו היא subscriptions. במקום אינטגרציה נפרדת לכל סוג אירוע (הודעה ב-Slack, CI-ב-PR, שעה ביום), יש framework אחד ו-tool אחד שבעזרתו ה-agent נרשם לאירוע ומתעורר כשהוא קורה.



דשבורד "Firings by shim": גרף עמודות מוערם של מספר ההפעלות של כל shim לאורך זמן. במקרא מופיעים shims כמו await_name_task_id ו-dynamic_tools_stringified_argume...

הרעיון הכי פרקטי בהרצאה הוא ניהול shims. shim הוא טלאי דטרמיניסטי על התנהגות רעה של המודל. למשל, אם Grok Bot לא עונה מספיק מהר, הוא מקבל דחיפה: "hey, the user's waiting on you". הם מעדיפים לתקן היום ולא לחכות ל-checkpoint הבא של המודל, אבל "we track all these things, which we call shims centrally". כל הפעלה נרשמת בלוג, bot עוקב אחרי הדשבורד, ו-"whenever a shim stops firing, that model or that agent will put up a PR and say, this shim can be cleaned up".

שתי מגמות נוספות: ה-harness עולה רמת הפשטה. הדור הקודם הורכב מקריאות LLM, ובדור החדש (Grok Bot וה-agent software factory) הוא אבן הבניין. והתחזית: "I predict that within six months, 90% of agent work is going to run off the laptop. So yesterday's harness may be dead, but long live the agent harness of tomorrow".

7. מסקנות ופעולות

- **בדקו את ה-harness שלכם:** מצאו הנחיות ו-guardrails שנועדו לפצות על חולשות קוד של מודלים ישנים, ומדדו אם הם עדיין משפיעים על התוצאה.
- **מדדו עלות לכל משימה, לא רק ציון:** שני המחקרים מראים שהבדלים בין harnesses מופיעים בעלות ולא באיכות.
- **רשמו כל טלאי כ-shim:** נהלו אותם במקום אחד, ספרו הפעלות, ומחקו shim שהפסיק להיות מופעל.
- **העדיפו primitives על פני אינטגרציות:** למשל מנגנון subscriptions אחד לכל האירועים שמעירים agent.
- **תכננו ריצה בענן:** משימות של שעות וימים לא ישרדו מכונה מקומית שנכנסת למצב שינה.
- **הגדירו הרשאות מצומצמות:** לכל agent ולכל אדם, עם guardrails נגד פעולות הרסניות וחריגות תקציב.

שאלה פתוחה	למה היא חשובה
כמה בדיוק חסכו ב-Grok Bot בעלות ובזמן תגובה, ואיך נמדדה האיכות?	הטענה המרכזית על יעילות לא גובתה במספרים מוגדרים.
איך נראית ארכיטקטורת ה-coordinator/worker בפועל?	זה ה-harness שלדבריו פותח את החזית הבאה, ולא ניתן עליו שום פרט.
על מה מבוססת תחזית ה-90% מחוץ ללפטופ?	זו הערכה אישית בלי נתונים.
האם המסקנה "ה-harness לא משנה" נכונה גם מחוץ ל-SWE-Bench Lite ו-MLBench?	שני ה-benchmarks מוגבלים ולא מייצגים עבודה ארוכה ורב-שלבית.
איך מאזנים בין פחות שקיפות למשתמש לבין אמון ובקרה?	המרצה ציין trade-off ולא הציע פתרון.

8. מילון מונחים

מונח	משמעות
Agent harness	השכבה שעוטפת את המודל, מכוונת אותו ומחברת אותו לכלים, למחשבים ולשירותים.
Runtime	שכבת היסוד: איפה רצה לולאת ה-agent ואיך נשמר state כדי שאפשר יהיה להמשיך עבודה.
Context / Orchestration	השכבה שקובעת איזה מודל נקרא ומתי ומה הוא רואה: prompts, tools, memory, compaction.
Compaction	דחיסת היסטוריית השיחה כדי שתיכנס בחלון ההקשר.
Data flywheel	מחזור שבו ה-harness מאפשר משימות, הדאטה מהן משפר את המודל, וה-harness מוחק את הטלאים.
Guardrails	כללים ובקורות שמגבילים או מתקנים את התנהגות ה-agent.
Shim	טלאי דטרמיניסטי זמני על התנהגות רעה של המודל, שנרשם בלוג ומוסר כשהוא מפסיק להיות מופעל.
Primitive	ממשק בסיס שאפשר להרכיב ולהרחיב לפיצ'רים, למשל subscriptions.
Subscriptions	מנגנון אחיד שבו agent נרשם לאירועים (Slack, CI, שעה) ומתעורר כשהם קורים.
Coordinator / Worker agents	agent מתאם ארון-חיים שמחלק משימות לצי של agents מבצעים.
Software factory	כמה agents שמכסים יחד את כל מחזור פיתוח התוכנה.
Meta harness	harness שמתזמר כמה agents ולא קריאות בודדות למודל.

מונח	משמעות
Hyrum's Law	כל התנהגות שנחשפת למשתמשים תהפוך בסוף לתלות שלהם.
SWE-Bench Lite / MLE-Bench	benchmarks להערכת agents במשימות קוד ובמשימות הנדסת ML.
Cache hit rate	שיעור הקריאות שבהן prefix של ה-prompt נשלף מה-cache, וזה מוזיל ומאיץ.