

איך לסמוך על ה-agents שלך: verification, skills וארכיטקטורה שבנויה ל-agents

Lauren Tan מסבירה איך היא הגיעה ממעקב צמוד אחרי כל agent ל-PRs שמתמזגים
אוטומטית, ולמה אמון הוא מה שקובע כמה אפשר להאיץ

Lauren Tan (מנחה לא מזוהה בוודאות) · מקור מקורי לא מזוהה (repost ב-X של Ricker) · הועלה 2026-10-08

משך: כ-55 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו הקלטה של שיחה בפורמט טקסט עם שיתוף מסך ושאלות מהצ'אט. המרצה היא Lauren Tan, שה- lower third, מציג אותה כ-SpaceX AI Software Engineer. בהקלטה היא מספרת שהצטרפה ל-Cursor לפני חמישה חודשים, שעבדה על Agents Window ושהיא מובילה את Grok Bot. מי שמנחה ומעביר את השאלות לא מוצג על המסך, וה- ASR קולט את שמו כ-Colin, כך שהשם לא ודאי. הסרטון הגיע כ-repost ב-X. הכיתוב שם ("35+ 2,000 PRs", "agents", "/pstack") לא מופיע בתמליל, והמספרים במסמך לקוחים מההקלטה עצמה.

הטענה המרכזית: מה שמגביל כמה agents אפשר להריץ במקביל הוא כמה אתה סומך עליהם, ולא המודל. האמון נבנה בשלוש שכבות. הראשונה היא verification, כלומר שה-agent מסוגל להריץ את האפליקציה ולבדוק בעצמו. השנייה היא skills שמלמדים אותו לעבוד כמו מהנדס אמיתי, ונבחנים ב-evals. השלישית היא ארכיטקטורה ו-CI שאוכפים כללים באופן קשיח, כך שהדרך הקצרה ביותר היא גם הנכונה. לדבריה, כך הגיעה למצב שבו ה-agents ממזגים PRs בעצמם והיא עוברת עליהם ב-main.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם timestamps, ומה לא נאמר.
- 2. **עקומת האמון:** המודל המנטלי של השיחה, אמון מול מספר agents, והמספרים שהיא מציגה.
- 3. **Verification ו-feature map:** ה-skill הראשון שבנתה ב-Cursor, ולמה מפה של הפיצ'רים היא מה שהפך אותו לשימושי.
- 4. **Skills, pstack ו-evals:** איך בונים skills מכשלים שרואים בפועל, ואיך בודקים אותם.
- 5. **מהמקומי לענן:** Cloud agents ו-Benny, ה-agent שמשחזר באגים.
- 6. **Dune:** ארכיטקטורה ו-CI קשיחים שנבנו בשביל agents, כולל מה בדיוק אסור.
- 7. **טוקנים, ROI ומי עוד שולח קוד:** שאלת העלות, ו-PMs ומעצבים שמגישים PRs.
- 8-9. **מסקנות ומילון:** פעולות מומלצות, שאלות פתוחות ומונחים.

הערות מקור המסמך נוצר אוטומטית מהתמליל ומהשקפים של ההקלטה. ה-ASR עלול לשבש שמות ומונחים: Grok Bot מופיע בתמליל כ-GrokBot, Rockbott, GrokBot ו-pstack כ-PSAC ו-P-Stack. בכל מקום כזה השתמשנו בכתיב שמופיע על המסך. הציטוטים מועתקים מילה במילה מהתמליל ונבדקו מול ה-timestamp שלהם. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

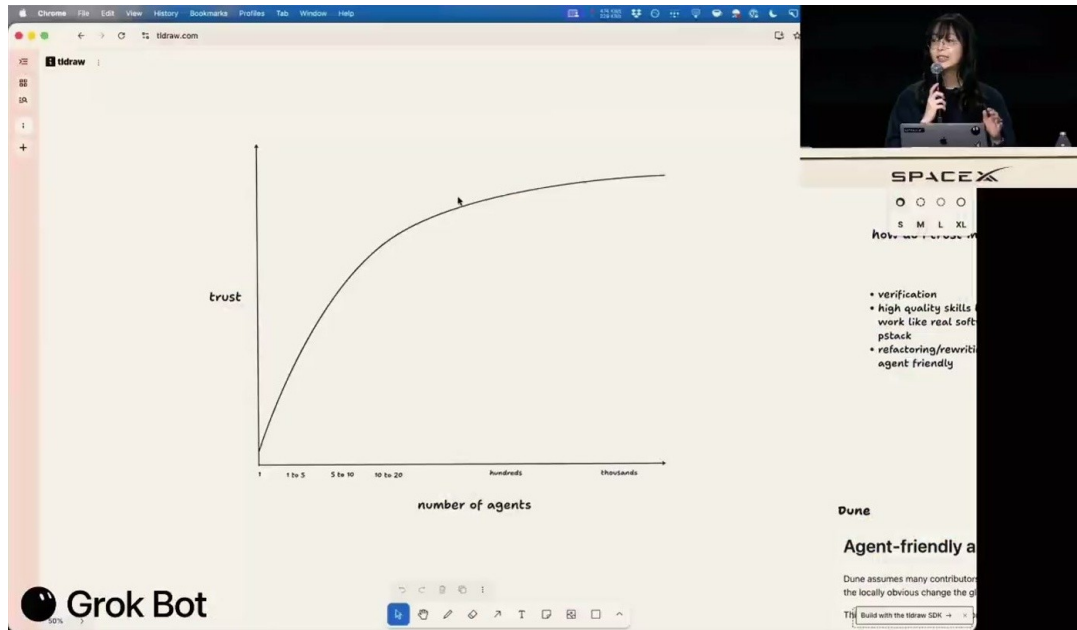
Tan מציגה את המעבר מעבודה צמודה עם agent אחד לעשרות PRs שמתמזגים בלי אדם כבעיה של אמון, ומשווה אותה לניהול צוות. מנהל שלא סומך על העובדים שלו נשאר במיקרו-מנג'מנט. מהנדס שלא סומך על ה-agents שלו נשאר צוואר הבקבוק. מה שבאמת מעניין בשיחה הוא הרשימה הקונקרטית של מנגנונים שבזכותם היא סומכת על ה-agents, יותר מהמספרים עצמם.

- **אמון הוא מה שמגביל, לא כמות ה-agents:** בלי אמון אי אפשר להרחיב לעשרות agents כשאתה לא סומך אפילו על אחד, ואתה עוקב אחרי כל פלט [3:00].
- **המספרים:** כאלף PRs בחודש שעבר, וכמעט 800 עד ה-12 לחודש הנוכחי [4:32]. בבוקר השיחה מצאה כ-20 PRs שכבר מוזגו ב-main [3:00]. ה-2,000 וה-35 agents מהכיתוב ב-X לא נאמרו.
- **Verification הוא ה-skill הכי חשוב:** ה-agent מריץ את האפליקציה, מצלם traces ו-heap snapshots ופותח simulator. זה לא מבטיח קוד טוב, אבל זה מבטיח קוד נכון [6:11].
- **Feature map הופכת את ה-verification לשימושית:** קובץ שמתאר כל פיצ'ר, איך מגיעים אליו מנקודת המבט של המשתמש, קיצורי מקלדת ו-selectors ל-[12:28]–[10:53] CDP.
- **Skills נבנים מתוך כשלים שרואים בפועל:** קוראים tool calls ו-thinking blocks, ומכל כשל חוזר עושים skill. ה-skills נבדקים ב-evals עם sub-agents שלא יודעים שבודקים אותם ועם judge ממודל אחר [18:32]–[21:40].
- **מתחילים מקומית, ורק אחר כך עוברים לענן:** מקומית אפשר לראות מה ה-agent עושה. Benny, ה-agent שמשחזר באגים בענן, מראה מה מקבלים בשלב הבא [23:15]–[24:47].
- **Greenfield שנבנה ב-vibe coding הוא הסיכון הגדול ביותר:** Grok Bot עבר refactor של יותר מ-600 PRs לארכיטקטורה חדשה בשם Dune, ורק אחרי זה היא הפסיקה לקרוא את הקוד [34:04].
- **אכיפה קשיחה עדיפה על כללים רכים:** useEffect אסור, הערות בקוד אסורות, וגבולות import בין main ל-renderer נבדקים ב-rules, skills ו-BugBot. הם רק שכבה משלימה [37:08]–[44:54].
- **טוקנים הם שאלה של ROI:** היא אומרת במפורש שהיא עובדת במקום עם טוקנים בלי הגבלה, וממסגרת את זה כהשוואה בין לגייס עוד אדם לבין להשקיע טוקנים בקוד שגם agent חלש מצליח לעבוד בו [47:58]–[49:29].

מה לא נאמר

- **אין מדד איכות:** לא נמסרו revert rate, שיעור באגים או regressions ב-PRs שמוזגו אוטומטית. היא עצמה אומרת שזה הוגן לשאול כמה מהקוד טוב [4:32].
- **גודל PR ממוצע לא ידוע:** טווח של 50 עד כאלף שורות, בלי נתון [35:36].
- **כמה agents במקביל ובאיזה מודל:** לא נאמר.
- **עלות:** אין מספר לטוקנים שה-refactor של 600 PRs צרך.
- **Dune לא יפורסם כקוד פתוח:** היא מתארת אותו כאוסף עקרונות ולא כמשהו שיפורסם [43:23]. pstack כן זמין לחיפוש.
- **תוצאות ה-evals:** לא הוצגו ציונים. יש רק תיאור של התהליך.

2. עקומת האמון

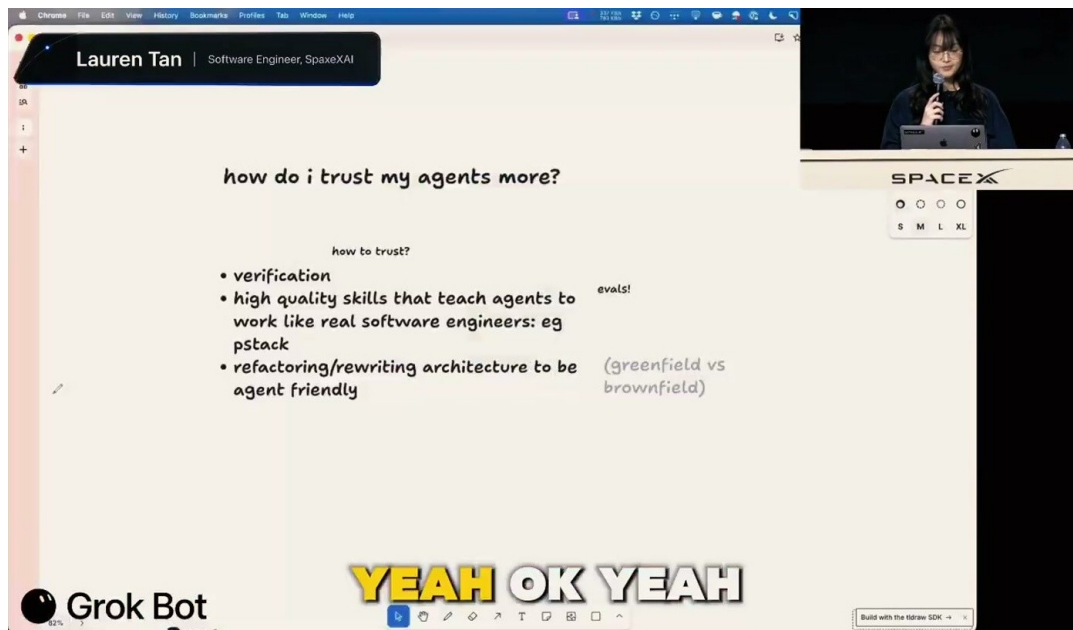


גרף שצוירה ביד: ציר Y הוא $trust$, ציר X הוא $number of agents$. לצד הגרף שלושת הכיוונים: $verification$, $high quality skills$ ו- $refactoring/rewriting architecture to be agent friendly$

לפני שנה, היא אומרת, רוב העבודה עם $agents$ הייתה צמודה: עוקבים אחרי כל פלט, כותבים $prompt$ אחרי $prompt$, ואין דרך להרחיב לעוד $agents$. בחמשת החודשים האחרונים היא "עלתה" על העקומה. לדבריה, גרף התרומות שלה ב-Cursor מתנהג כמו עקומת האמון: בחודש הראשון היא כמעט לא הייתה פרודוקטיבית כי למדה את ה-codebase, ומאז הקצב עלה בחדות.

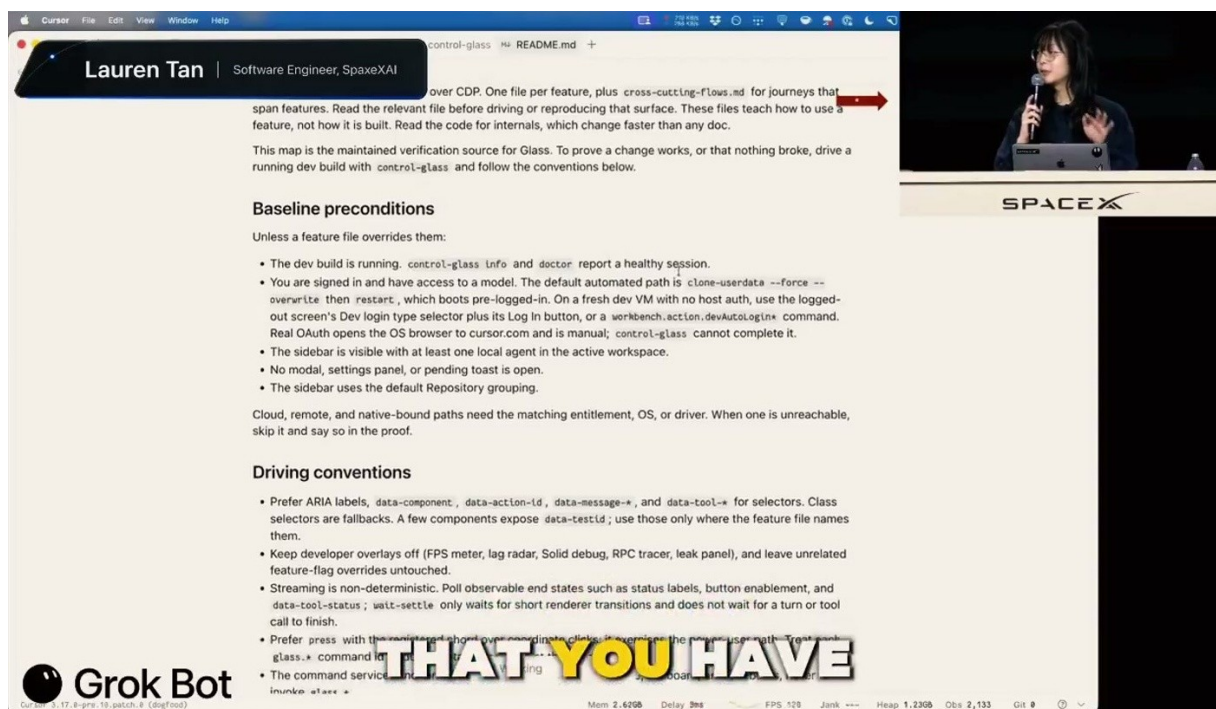
המספרים שנאמרו: "last month I shipped a thousand PRs which is ridiculous and then this month we're only on the 12th I'm already at like almost 800 PRs landed" [4:32] ייצור מסה של קוד גרוע: "this sounds kind of scary to say this and it makes me sound like a slop artist" [3:00] but I promise I'm not" [3:00] המסר המעשי הוא שאין קיצורי דרך על העקומה. מי שעדיין בתחתית לא צריך להפעיל מאה $cloud agents$, כי "you're just going to waste a lot of tokens and it's going to be extremely expensive" [26:18].

— Lauren Tan [27:49] I think there's really no shortcut for going from here to there, because this is really about your personal level of trust in agents



השקף "how do i trust my agents more?": הרשימה verification, high quality skills that teach agents to work like real software engineers (eg pstack), refactoring/rewriting architecture to be agent friendly (greenfield vs brownfield) ו-

3. Verification ו-feature map

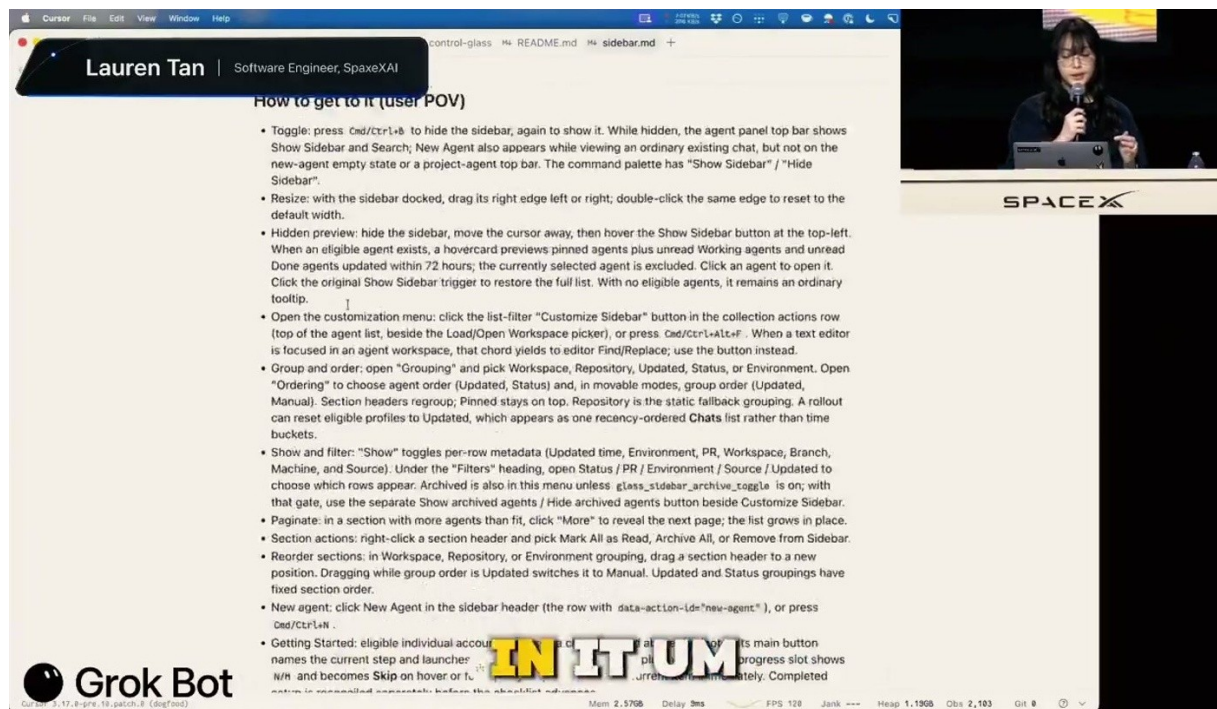


README של ה-control-glass, שמשמש לשליטה ב-dev build של Agents Window: סעיפי Baseline preconditions ו-driving conventions

ב-verification היא מתכונת לכך שה-agent מסוגל לעשות כל מה שמשמש עושה עם האפליקציה: להריץ אותה, לצלם CPU traces ו-heap snapshots, לפתוח iOS simulator. זה מה שסוגר את הלולאה. הסיפור שלה מהשבוע הראשון ב-Cursor: Agents Window עמד לצאת בעוד שבוע, והיא ישיבה מול flame graph ב-Chrome DevTools, צילמה מסך ושלחה ל-agent. ה-agent קבע בביטחון מה הבעיה, והתברר שזו לא הבעיה. "you are the verifier, right? You're the bottleneck" [7:46]

לכן ה-skill הראשון שבנתה הוא control-glass. השם הפנימי של Agents Window. ה-skill מלמד את ה-agent לשלוט ב-Electron app דרך Chrome DevTools Protocol. היא אומרת שהקוד עצמו לא מעניין, ושה-agent יכול לכתוב אותו לבד. החלק החשוב הוא קובץ ה-feature map. בלי המפה, ה-agent ידע להפעיל את

האפליקציה אבל לא ידע איפה נמצא כל פיצ'ר. כשמישהו דיווח שה-sidebar השמאלי איטי, ה-agent חיפש בקוד ולחץ באקראי, והתוצאה הייתה "just an awful experience" [10:53].



קטע מה-feature map: רשימת Sub-features של ה-sidebar וסעיף How to get to it (user POV), עם הוראות ניווט ופעולות

המפה מתארת כל פיצ'ר, את תתי-הפיצ'רים שלו, איך מגיעים אליו מנקודת המבט של המשתמש, קיצורי מקלדת, ואת ה-attributes ב-DOM שמשמשים לבחירת אלמנטים דרך [12:28] CDP. בזכות המפה ה-agent יכול לטפל גם בדיווח גרוע: ב-Slack הפנימי של Cursor מגיעים לא פעם צילום מסך ושלושה סימני שאלה, ועם המפה ה-agent יודע לאן לנווט. ב-pstack יש skill בשם create verification שבונה מפה כזו מתוך סריקה של הקוד, ו-skill בשם maintain verification שמעדכן אותה.

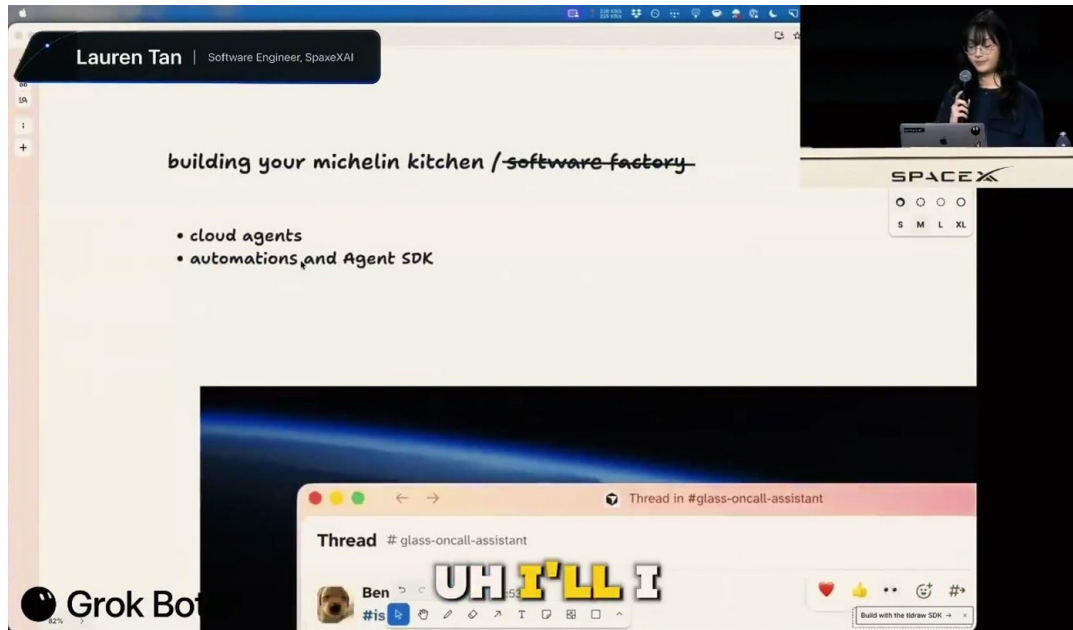
4. Skills, evals-ו

pstack (שם עם קריצה: "potato snack") הוא plugin שלה ל-Cursor, בהשראת GStack של Garry Tan מ-Y Combinator. לדבריה אין ביניהם קשר משפחתי [13:59]. היא לא תכננה לבנות אותו. הוא נבנה אחרי skill מתוך תצפית על כשלים. דוגמה: ה-agent קבע את שורש הבאג בביטחון, וכשבדקה את ה-tool calls התברר שהוא בכלל לא קרא את הקוד הרלוונטי. מזה נולד skill שבעיקרו אומר "קט stop hallucinating actually go and search up look up the code use a lot of subagents" [15:30]. במשל הניהול שלה, ה-agent הוא מהנדס מצוין שגויס לפני חמש שניות ואין לו שום הקשר עסקי, ו-skill הוא ה-onboarding שלו.

כדי לבדוק skill היא משתמשת ב-evals, שהיא מגדירה כ-"it's like a unit test for an agent" ב-pstack. יש agent eval playbook מתאם כותב rubric, מפעיל sub-agents, כל אחד בתיקיה עם שם שלא מסגיר שמדובר במבחן, כי "agent" [18:32]. agents can actually tell and when they do they change their behavior" [21:40]. בגלל ש-Cursor תומך במודלים רבים, אפשר להריץ את אותו eval על כמה מודלים. את אותו תהליך היא הפעילה גם על ה-skill של ה-verification.

Lauren Tan [20:09] But I will say like maintaining skills is actually pretty hard. It requires, I think, a — lot of taste and observation

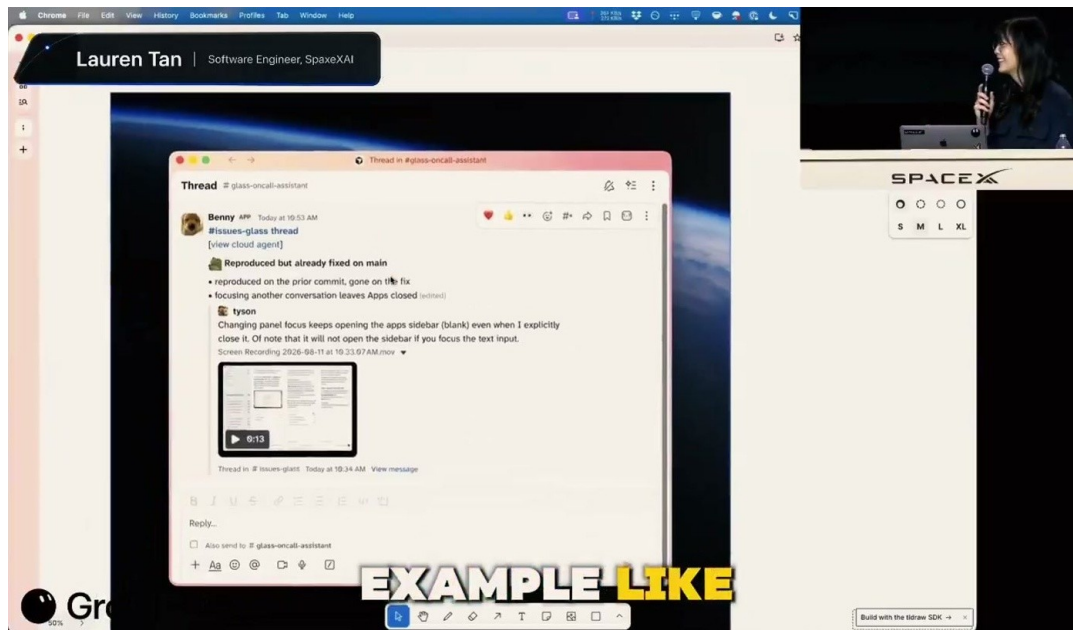
5. מהמקומי לענן: Benny-cloud agents



השקף "building your michelin kitchen / software factory" (המילים software factory מחוקות): cloud agents, automations and Agent SDK, וליצידו thread ב-Slack מ-Benny

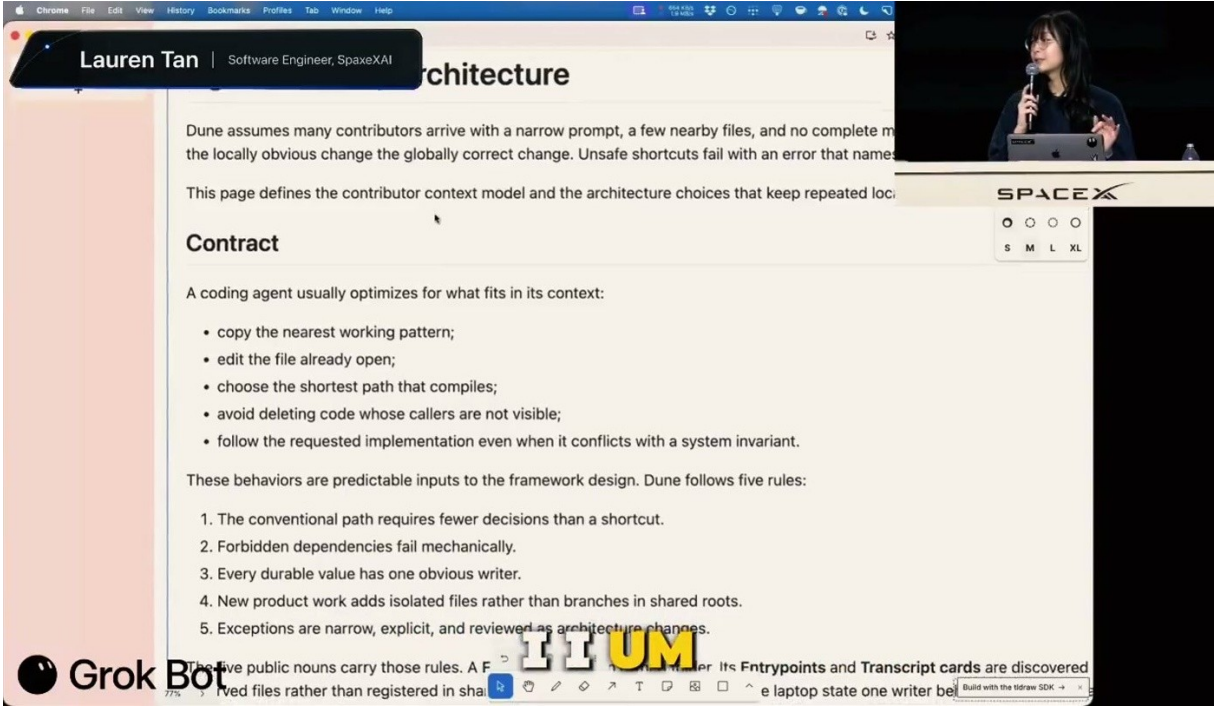
מה עושים בפועל? [23:15] "I think that the best place to start is local because you can observe" . מקומית רואים איך ה-agent משתמש באפליקציה ובאילו APIs הוא קורא. אחרי שיש אמון, הערך עובר לענן: skill של verification שהוגדר פעם אחת משפר את כל הצוות ואת כל החברה, לא רק מהנדס בודד.

הדוגמה היא Benny, agent שמקבל את כל דיווחי הבאגים, מריץ שם את Cursor עם אותם control skills ומנסה לשחזר. בדוגמה שהוצגה, Benny שחזר את הבאג וגילה שהוא כבר תוקן ב-main, כך שכל מה שנשאר הוא לשחרר build. לדבריה זה חוסך שעה של בירור ידני מול [24:47] agent. המשל שלה: המהנדס הוא השף הראשי במטבח. הוא לא מבשל בעצמו, הוא מתכנן את המטבח ומחלק משימות [21:40].



thread בערוץ issues-glass ב-Slack: הדיווח של "Reproduced but already fixed on main", Benny, עם סיכום הצעדים וצילום מסך

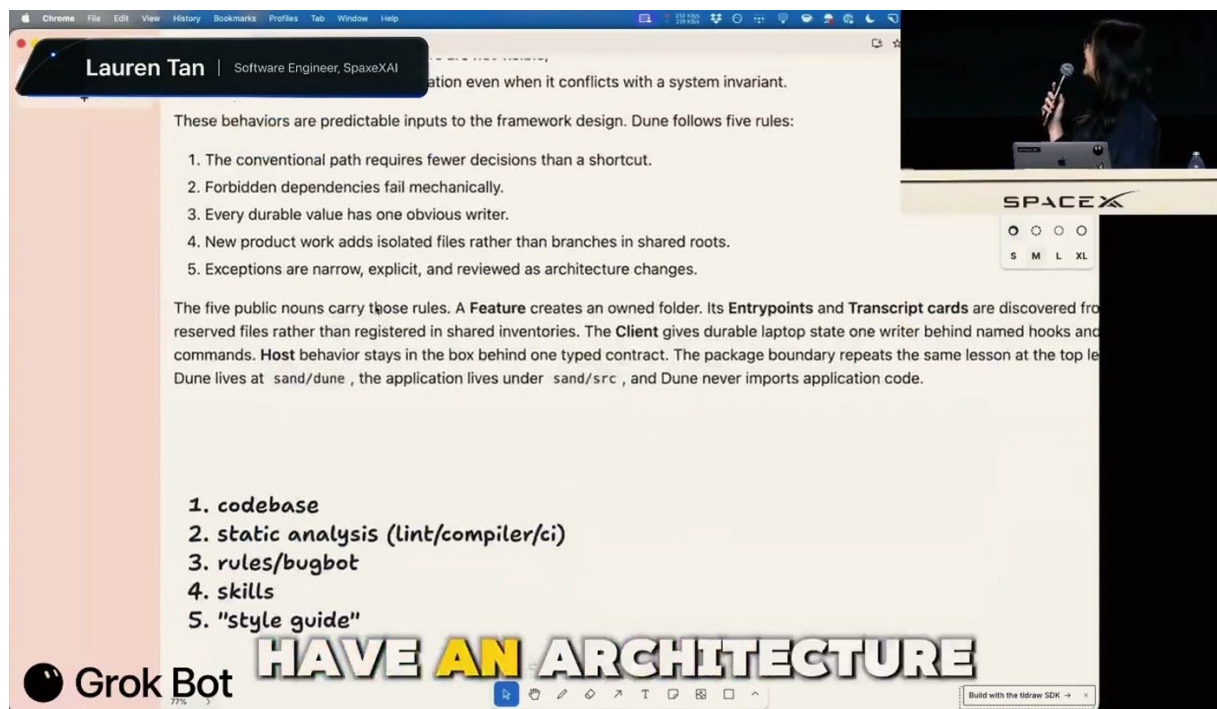
Dune .6: ארכיטקטורה שנבנתה בשביל agents



ומסיף Contract עם רשימת הדברים ש-coding agent נוטה לעשות כשהוא מייעל לפי מה שנכנס לו ל-context *"Dune, Agent-friendly architecture: "Dune assumes many contributors arrive with a narrow prompt, a few nearby files* מסמך

החלק השלישי בשיחה עוסק ב-refactoring וכתובה מחדש. ב-big tech, למשל ב-Meta, התשתית בנויה בשביל המהנדס הכי פחות מנוסה בצוות: frameworks, קונבנציות והרשאות מוגבלות. Codebase כזה (brownfield) כבר מתאים ל-greenfield. agents שנבנה ב-vibe coding הוא "like the biggest risk in my opinion and also the greatest opportunity" [30:57]. כך התחיל Grok Bot, אפליקציה שהושקה יום לפני השיחה ומאפשרת לתזמר agents שלכל אחד מהם זהות משלו. בהתחלה אף אדם לא קרא שם קוד, וה-agents פתרו כל משימה בדרך הנוחה ביותר. היא קוראת לזה organic architecture.

הפתרון היה refactor של "over 600 PRs" ל-Dune, ארכיטקטורה שהיא מתארת כ-Next.js ל-Electron שנכתב בבשבייל [34:04] agents. העיקרון המנחה: "agents" [43:23] "the shortest path is the best path" תמיד ילכו בדרך הקצרה, אז צריך לדאוג שהדרך הקצרה תהיה הנכונה. כל feature יושב בתיקיה אחת, ויש שמות עצם קבועים ב-b-framework: feature, entry points, transcript cards. בגלל זה כ-80% מהעבודה נשאת בתוך תיקייה אחת.



חמשת הכללים של Dune (למשל Forbidden dependencies fail mechanically, Every durable value has one obvious writer ומתחתם חמש שכבות האכיפה: codebase, static analysis (lint/compiler/ci), rules/bugbot, skills ו-"style guide").

- **useEffect אסור:** "one of the biggest foot guns in react is use effect". ב-Dune CI נכשל עליו [37:08].
- **הערות בקוד אסורות:** לדבריה, ב-99% מהמקרים agents כותבים הערות על היסטוריה לא רלוונטית, למשל ש-Lauren אמרה פעם שאסור לעשות משהו [38:42].
- **גבולות main/renderer:** ב-Electron כל frame צריך להיגמר תוך 16ms כדי להגיע ל-60FPS. ב-Agents Window, קוד כבד שנגרר ל-renderer גורם לקפיצות. ב-Grok Bot יש תיקיות electron main ו-electron renderer, ובדיקת import ב-CI עוברת על גרף התלויות [40:14].
- **שכבות רכות כתוספת בלבד:** rules, BugBot, skills ו-style guide הם שכבות 3 עד 5. ה-agent יכול לשכוח אותם, ולכן אסור שיהיו מנגנון האכיפה היחיד [44:54].
- **בחירת שפה:** Rust חוזרת לפופולריות בגלל ה-compiler הקשוח וה-borrow checker, כל עוד ה-agents לא כותבים unsafe [44:54].

Lauren Tan [46:26] Every time you have to do that, you should consider that as a code smell, like — a anti-pattern. And you should say, okay, instead of me commenting on the PR, how do I turn this into ?a hard rule, right? How do I turn this into a lint rule

7. טוקנים, ROI ומי עוד שולח קוד

שאלה שחזרה בצ'אט כמה פעמים: האם זה ריאלי בלי תקציב טוקנים בלי הגבלה? היא עונה ישירות: "I work at an AI lab where we have unlimited tokens" [47:58]. עם זאת, היא טוענת שאפשר להגיע לשם בלי לפשוט רגל, ושהשאלה היא ROI. המחיר הגבוה משולם בהתחלה, על ה-refactor ועל בניית ה-guardrails. החלופה היא לגייס עוד אדם או לגדול לארגון הנדסה של עשרת אלפים איש. אחרי ההשקעה, גם agents קטנים כותבים קוד טוב [49:29]. בהמשך היא מזכירה שבאותו יום הוכרז Grok 4.6, ומוסיפה בזירות של מיטב ידיעתה המחיר לטוקן זהה ל-4.5 [51:00].

לגבי צוות המוצר: Grok Bot, שהממשק שלו נראה כמו iMessage, הוא לדבריה "רגע ה-Cursor" של אנשים שלא באים מהטכנולוגיה. PMs משתמשים בו גם כדי לקבל סיכום של מה ש-Lauren עשתה בלילה, והם גם מגישים תיקונים. היא סוקרת ומאשרת. "I think that shows that, you know, the Dune architecture is holding up" [54:04].

8. מסקנות ופעולות

- בנו skill של verification לפני כל דבר אחר: שליטה באפליקציה דרך CDP, simulator או CLI, יחד עם feature map שמתעדכנת. pstack כולל skills מוכנים ל-create ול-maintain.
- תעדו כשלים מתוך tool calls: קראו thinking blocks ו-tool calls. כל כשל שחוזר הופך ל-skill, וכל skill נבדק ב-eval עם judge ממודל אחר.
- הפכו הערות ב-code review לכללים: כל הערה ידנית חוזרת היא סימן ל-lint rule, לבדיקת CI או לשינוי ארכיטקטורה שחסרים.
- התחילו מקומית: שניים או שלושה agents במקביל, עם צפייה. ל-cloud agents ולאוטומציות כמו Benny עוברים רק אחרי שיש אמון.
- תכננו כך שהדרך הקצרה תהיה הנכונה: feature בתיקייה אחת, תלויות אסורות שנכשלות באופן מכוני, ו-CI שמעצבן בני אדם וש-agents סופגים בלי בעיה.

9. מילון מונחים

מונח	פירוש
Verification	היכולת של ה-agent להריץ את המוצר בעצמו ולבדוק שהשינוי עובד
Skill	קובץ markdown עם הוראות שמלמד את ה-agent דרך עבודה
Feature map	קובץ שממפה כל פיצ'ר: איך מגיעים אליו, קיצורים ו-selectors
Eval	מבחן ל-skill או ל-agent, כמו unit test
Judge agent	agent ממודל אחר שבודק את הציונים כדי לנטרל הטיה
Hill climbing	הרצה חוזרת עד שמגיעים לציון היעד
CDP	Chrome DevTools Protocol, ממשק לשליטה תכנותית בדפדפן או ב-Electron
Cloud agents	agents שרצים בסביבה מרוחקת, כל אחד במכונה משלו
Auto-merge	מיזוג PR בלי אישור אנושי מוקדם
Greenfield / Brownfield	מערכת חדשה מאפס לעומת מערכת קיימת עם תשתית
Guardrails	מגבלות שמונעות מתורמים, בני אדם או agents, לגרום נזק
Renderer / Main process	התהליך שמצייר את ה-UI ב-Electron, לעומת תהליך הרקע
BugBot	כלי ה-code review של Cursor שרץ ב-CI
Organic architecture	מבנה קוד שנוצר בלי תכנון מכוון, כש-agents בוחרים בכל פעם את הדרך הנוחה