

Grok Bot: צוות של agents אוטונומיים שמנהל את ה-coding agents בשבילך

workshop מ-Grok Bot Galaxy: איך מהנדס אחד מנהל fleet של bots ש-מפעילים,
בודקים ומכווינים Cursor cloud agents, גם כשהוא לא ליד המחשב

Lingxi (שם לא ודאי; בתמלול: Ling Shi) · Grok Bot Galaxy Day 1 (לפי השקופיות; הסרטון הגיע מ-repost ב-X) ·
2026-10-04

משך: כ-45 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה סיכום של workshop מהיום הראשון של Grok Bot Galaxy, כך זה מופיע בתחתית השקופיות. הדובר מציג את עצמו כמהנדס תוכנה ב-SpaceX AI. לדבריו הוא הצטרף ל-"Cursor, which is now SpaceX AI", לפני תשעה חודשים, ומאז הספיק לשחרר שני מוצרים. הוא בונה את Grok Bot, ולדבריו בחודשיים האחרונים הוא עובד רק איתו. השם שלו לא ודאי: בתמלול הוא Ling Shi, ובכיתוב של ה-repost הוא Lingxi Li. גם ה-bot שמשמש אצלו chief of staff נקרא Lingxixi.

הטענה המרכזית: אחרי agentic coding, autocomple, ו-cloud agents, הצעד הבא הוא שכבה של agents אוטונומיים עם שם, זיכרון ו-routines, שעובדים כמו עמיתים. הם כותבים את ה-prompts ל-coding agents, עוקבים אחריהם, קוטעים אותם כשהם סוטים, בודקים שצורפו הוכחות ומעבירים ידע זה לזה. בחצי השני של ה-workshop יש דמו חי של fleet כזה על אתר הדגמה בשם Flylo.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות בשורה אחת כל אחת, ומה לא נאמר.
- **2. עקומת הבשלות:** איך הדובר ממקם את Grok Bot ביחס ל-autocomple, ו-cloud agents.
- **3. מה זה Grok Bot:** ארבעת הרכיבים, ובמה זה שונה לטענתו מכלים אחרים.
- **4. שימושים:** audit, Slack, לילי, internal tooling ו-autofix ל-CI.
- **5. ארגון ה-fleet והדמו:** מי ה-bots, למה מפצלים ביניהם, onboarding בין bots, הגדרת P0 ו-playbook.
- **6. מה לקחת מכאן:** שלושת הלקחים שבסוף, צעדים מעשיים ושאלות פתוחות.
- **7. מילון מונחים:** המונחים באנגלית שמופיעים במסמך.

הערות מקור המסמך הופק אוטומטית מתמלול ההקלטה ומהשקופיות שלה. השמות והמונחים עלולים להיות משובשים בגלל זיהוי הדיבור, למשל Grok Bot מופיע בתמלול גם כ-GropBot, Grubbot ו-ScrocBot, ושם הדובר לא ודאי. הציטוטים מועתקים מילה במילה מהתמלול ובבדקו מול חותמת הזמן שלהם. לא נוסף שום מידע ממקורות חיצוניים. הכיתוב של ה-repost ב-X לא שימש מקור לעובדות.

1. תקציר מנהלים

ה-workshop הוא חצי הרצאת מוצר וחצי הדגמה של זרימת עבודה אישית. הדובר לא מציג benchmark ולא נתונים. הוא מציג ארכיטקטורה ארגונית של bots: chief of staff, שמנתב, bots לפי תחום ו-playbook משותף. לצד זה הוא מראה סדרה של routines שמחליפות את האדם במעקב אחרי ה-coding agents ובבדיקת התוצרים שלהם.

- **הצעד הבא אחרי cloud agents הוא agent שמנהל agents:** הדובר מספר שכבר בניהול של Cursor 15 cloud agents במקביל הוא קפץ כל הזמן ביניהם. Grok Bot נועד לכתוב את ה-prompts, לשלוח follow-ups, לנהל תור הודעות, לקטוע ולהכווין [3:02].

- **האינטגרציה first-party עם Cursor היא היתרון המרכזי:** ה-bot קורא transcripts, מפעיל cloud agents ומגיב להם דרך tool calls, לא דרך ממשק. לפי הוראות שהוגדרו לו מראש הוא גם בודק שצורפו הוכחות, למשל screenshots או השוואת ביצועים לפני ואחרי [4:34, 6:05].

- **רץ 24/7 על מחשב משלוי:** "9:17" [9:17] "you don't need any caffeinated laptop anymore". הוא זמין ב-iOS, iPad, Android, Windows, Linux, macOS, ואפשר להשתלט מרחוק על המחשב שלו כדי לעבור login או אימות אנושי בלי למסור סיסמה [10:48].

- **זיכרון נפרד לכל bot:** לכל bot יש שם וזיכרון משלו, כך שמספיק לתת הוראה פעם אחת והיא נכנסת מעצמה ל-prompts הבאים שהוא שולח ל-cloud agents [7:38].

- **הלילה וה-on-call עוברים ל-bots:** לילי פותח PRs לניקוי slop ולבעיות security. מטפל ב-CI אדום, ו-on-call מקבל ping רק אם הבעיה לא נפתרה תוך כ-10 דקות [16:57, 21:41].

- **fleet לפי תחומים במקום bot אחד:** יש chief of staff ו-lcraig ל-Steve, UI ל-DevEx ו-lHogan ל-Infra. לפי הדובר הפיצול נובע ממגבלות context ומ-pipeline שונה לכל תחום, לא מהיכולת של המודל [21:41].

- **ידע ארגוני עובר מ-bot ל-bot, לא בהעתק-הדבק:** בדמו bot ותיק עושה onboarding ל-bot חדש מ-Jenny. marketplace, head of operations, מחזיקה playbook ב-Notion ומפיצה אותו לשאר ה-bots [24:53, 37:17].

- **דחיפות מוגדרת כנוהל, לא כמילה:** הגדרת P0 היא routine שבודק את ה-cloud agents כל חמש דקות וקוטע אותם כשהם תקועים על sleep 300 או זהירים מדי [32:37, 34:07].

- **העבודה של האדם עוברת לבדיקת הוכחות:** "I don't really check my cloud agents I check the results" [38:51] "and check the proof". מכאן הדרישה ב-playbook שלכל PR יצורפו screenshots או perf metrics.

מה לא נאמר

- **אין מספרים מדויקים:** הטענה שכל צוות Cursor הפך ל-[3:02] 10x, והאמירה ש-"most of the time" הבעיה נפתרת תוך 10 דקות [21:41], נאמרו בלי שום מדידה.

- **אין עלות:** לא נאמר כלום על המחיר של bots שרצים 24/7 ומפעילים עשרות cloud agents.

- **אין פרטים על auto-merge:** הדובר מתאר merge אוטומטי לפי תנאים, אבל לא מסביר אילו מנגנוני הגנה יש, איך עושים rollback, ומה שיעור ה-PRs הגרועים.

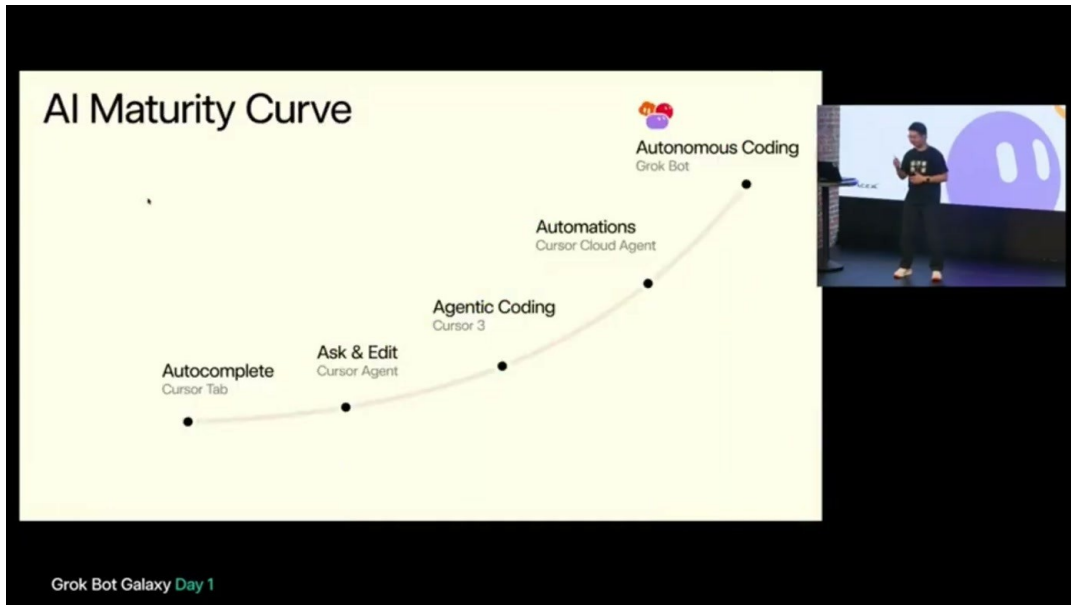
- **אין פירוט על הרשאות:** על TestFlight נאמר רק [20:02] "in a safe way". לא הוסבר איך מנהלים credentials ל-bots עם גישה לכל הכלים.

- **המודל לא נחשף:** נאמר רק שכל ה-[21:41] "sitting on the same model" bots.

- **הדמו לא הסתיים:** לא הוצג אם התקלה ב-Flylo תוקנה ונמזגה. ה-PR עם ה-screenshots שהוצג בסוף הגיע agent-מ שהופעל לפני ההרצאה [41:57].

- **ההשוואה למתחרים לא מבוססת:** הדובר אומר ש-OpenClaw ו-Hermes agent לא היו טובים בעבודה הנדסית, בלי להציג נתונים [9:17].

2. מ-autocomplete ל-fleet: עקומת הבשלות



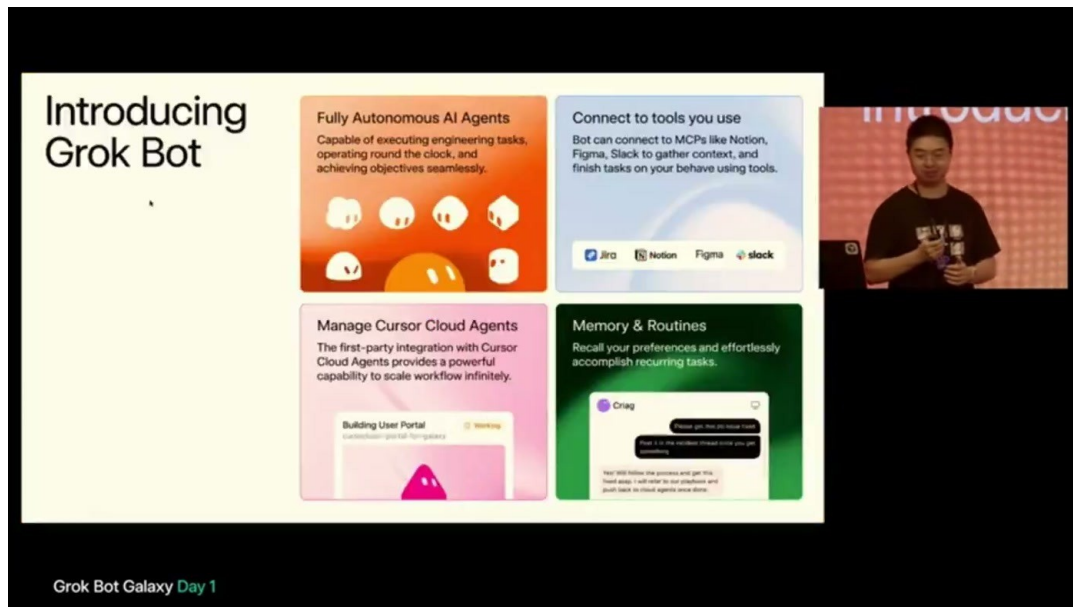
שקופית AI Maturity Curve: עקומה עולה מ-Autocomplete (Cursor Tab), Ask & Edit (Cursor Agent), Agentic Coding (Cursor 3), Automations (Cursor Cloud Agent), עד Autonomous Coding (Grok Bot)

הדובר פותח ברקע אישי. בתשעה חודשים ב-Cursor הוא בנה את "Cursor 3 Asian Window" (כנראה שיבוש של Agents Window, לא מאומת) ועכשיו הוא בונה את Grok Bot. לדבריו את הגרסה הראשונה של אפליקציית המובייל של Grok Bot הוא בנה לבד בשלושה שבועות [0:00].

אחר כך הוא עובר על השלבים. הכול התחיל ב-autocomplete תחבירי, והמשיך ב-Ask & Edit. Tab completion. Ask & Edit. היה הפעם הראשונה שה-agent יכל לעבור על ה-codebase. ב-Cursor 2 וב-3 הגיע agentic coding: שליטה בעכבר ובמקלדת לבדיקות end-to-end, תכנון לטווח ארוך ופקודות כמו goal/ ו-loop [1:30]. אחריו הגיעו automations על Cursor cloud agents: הפעלה לפי triggers כמו הודעת Slack, ריצה 24/7, ועשר מכונות ומעלה במקביל, כך שה-agents כבר לא "fight for the same port" [3:02].

הפער שלטענתו נשאר: עדיין צריך לכתוב prompt, להפעיל agent ולדחוף אותו. "I was able to manage 15" [3:02] cursor cloud agents, and it was already a lot", הוא רצה agent נוסף שישלח follow-ups, ינהל תור הודעות ויקטע agent כשצריך לשנות כיוון. החיבור של זה עם זיכרון ותקשורת ישירה הוא Grok Bot [4:34].

3. מה זה Grok Bot ולמה לא עוד agent



שקופית Introducing Grok Bot עם ארבעה כרטיסים: Fully Autonomous AI Agents, Connect to tools you use (עם לוגואים של Notion, Figma ו-Slack), Manage Cursor Cloud Agents, ו-Memory & Routines

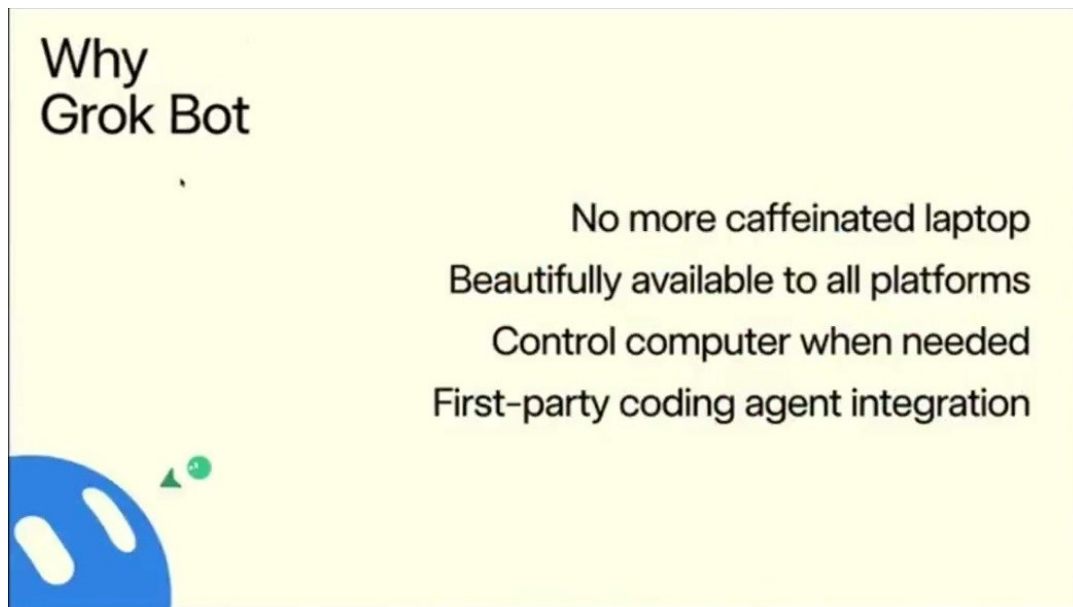
ארבעת הרכיבים, כמו שהדובר מתאר אותם:

אוטונומיה: ה-bot רץ 24/7, לא תלוי במחשב הנייד שלך ולא צריך אותו בכלל [4:34].

ניהול coding agents: זה לדבריו "the biggest unblock". הוא יכול לנהל כמעט כל coding agent, ול-Cursor cloud agents יש לו אינטגרציה first-party. הוא קורא transcripts, מפעיל agent, ויכול להריץ גם על private worker כמו Mac משלך. כל מה שאפשר לעשות בחלון של Cursor הוא עושה דרך tool calls. אחרי שזרימת עבודה מסתיימת, הוא בודק לפי ההוראות שלך אם צורפו הוכחות. אם חסרות, הוא שולח follow-up ל-cloud agent בעצמו [6:05].

חיבור לכלים: למשל Vercel MCP. ההבדל לעומת agent מקומי הוא שה-bot שומע את ה-signal כשה-build נכשל ומתחיל לעבוד לבד. אפשר גם לבקש ממנו deploy ב-6 בבוקר מחר [6:05, 7:38].

Memory ו-routines: לכל bot יש שם וזיכרון נפרד, כדי שלא יוצף בזיכרונות לא רלוונטיים. מה שלימדת אותו נכנס ל-prompts הבאים שהוא שולח ל-cloud agents [7:38].



שקופית Why Grok Bot עם ארבע שורות: No more caffeinated laptop, Beautifully available on all platforms, Control computer when needed, First-party coding agent integration

- **בלי laptop ער:** הדובר מתייחס לטריקים שמשאירים את הנייד ער כדי שמשימה תסתיים. לדבריו זה לא אלגנטי, ול-Grok Bot יש מחשב משלו 24/7 [9:17].
- **כל הפלטפורמות:** לאחרונה יצאו גרסאות ל-iPad ול-Android, בנוסף ל-Linux, Windows, ו-iOS, ו-macOS [9:17].
- **שליטה במחשב כשצריך:** כשה-bot נתקע על login או על אימות אנושי, אפשר להיכנס למחשב המרוחק מהמובייל או מה-desktop וללחוץ במקומו, בלי למסור לו סיסמה [10:48].
- **אינטגרציה first-party:** לדבריו זה קיים רק ב-Cursor. הוא מספר שהשתמש הרבה ב-OpenClaw וב-Hermes agent, ושהם לא היו טובים בעבודה הנדסית בגלל שחסרה להם אינטגרציה כזו [9:17].

4. שימושים: מה ה-bots עושים כשאינן אחד מול המחשב

מקרה הבסיס: אתה בטיסה או ישן, ואתה ה-code owner היחיד של service שחבר צוות צריך שיאשרו לו. ה-bot מנטר את Slack, בודק לפי הקריטריונים שהגדרת, משחרר את החסימה ומדווח בחזרה. הצעד הבא הוא לתת לו את קריטריוני ה-review שלך, למשל screenshot מצורף ו-"that are not fake tests". מעכשיו כל תיוג ב-Slack מפעיל skill של code scan בתוך cloud agent שרץ ב-[12:20, 13:55] repository. צעד נוסף: ניטור של דיווחי באגים על Grok Bot ב-X. ה-bot בודק אם הבאג משתחרר, ואם כן הוא מתקן ושולח [13:55] PR. מה נשאר לאדם: כיוון המוצר, פרטי עיצוב, בעיות ביצועים וארכיטקטורה, והכי חשוב, ללמד את ה-bot להגיד לא.

— Lingxi [15:27] The bot is very kind. They will possibly accept everything by default, so you need to set the boundary.

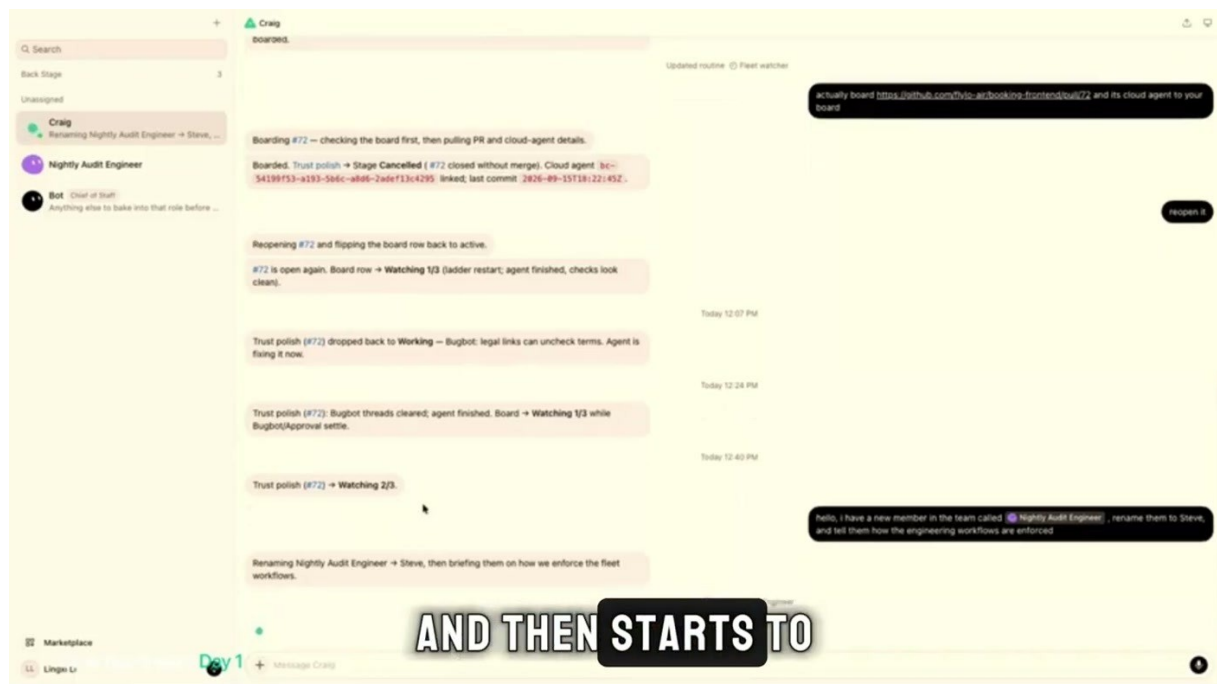
- **Nightly code cleanup:** תבנית שזמינה ב-marketplace. בלילה אף אחד לא דוחף קוד, אז הסיכון ל-conflict נמוך, והשינויים עצמם בסיכון נמוך: לאחד slop למודול קטן יותר, לקצר הערות ארוכות. ה-cloud agent יכול למזג לבד אם צורפה הוכחה end-to-end. אצל הדובר זה רץ על כל ה-monorepo: איכות קוד, מודולריזציה, הערות ו-security audit. בבוקר מחכים לו [16:57, 18:28] PRs. בשעה יש סתירה: בהתחלה נאמר 3 AM [16:57] ובדמו "Usually, it runs on 4 AM" [26:25].
- **Internal tooling בלי dashboard:** את ההוספה של אנשים ל-TestFlight לפי כתובת מייל הוא לא בנה כאפליקציה. מתייגים את ה-bot ב-Slack עם כתובת המייל, ו-routine שמופעל מ-webhook של Slack מבצע את ההוספה. לדבריו מספיקים prompt אחד וגישה בטוחה ל-[18:28, 20:02] TestFlight.
- **Autofix לכל דבר:** כשה-CI אדום, ה-bot בודק מה נכשל, מפעיל cloud agent שמתקן, וממזג לפי הקריטריונים שהגדרת. זה תקף גם ל-deployment errors, ל-CI flakiness, ולכל alert. ה-on-call מקבל ping רק אם לא נפתר תוך כ-10 דקות [20:02, 21:41].

• **Bots לא נחמדים:** הדובר מדגיש שה-bot צריך חוויה טובה כשאדם צריך לשחרר אותו בשלב authentication authentication [15:27, 16:57].

5. ארגון ה-fleet ודמו חי

הערה	תפקיד	Bot
מנתב בקשות, יודע מי עובד על מה, ולא ממלא את ה-context שלו בפרטים הנדסיים	Chief of staff	Lingxixi
בדמו הוא ה-engineer bot הוותיק, והדובר קורא לו גם head of engineering	UI engineering	Craig
בדמו זה ה-nightly audit engineer מה-marketplace שקיבל את השם Steve	DevEx	Steve
נזכר רק בהסבר על הפיצול	Infra	Hogan
נוספה בדמו, אחראית על ה-playbook ב-Notion ומפיצה אותו ל-bots	Head of operations	Jenny

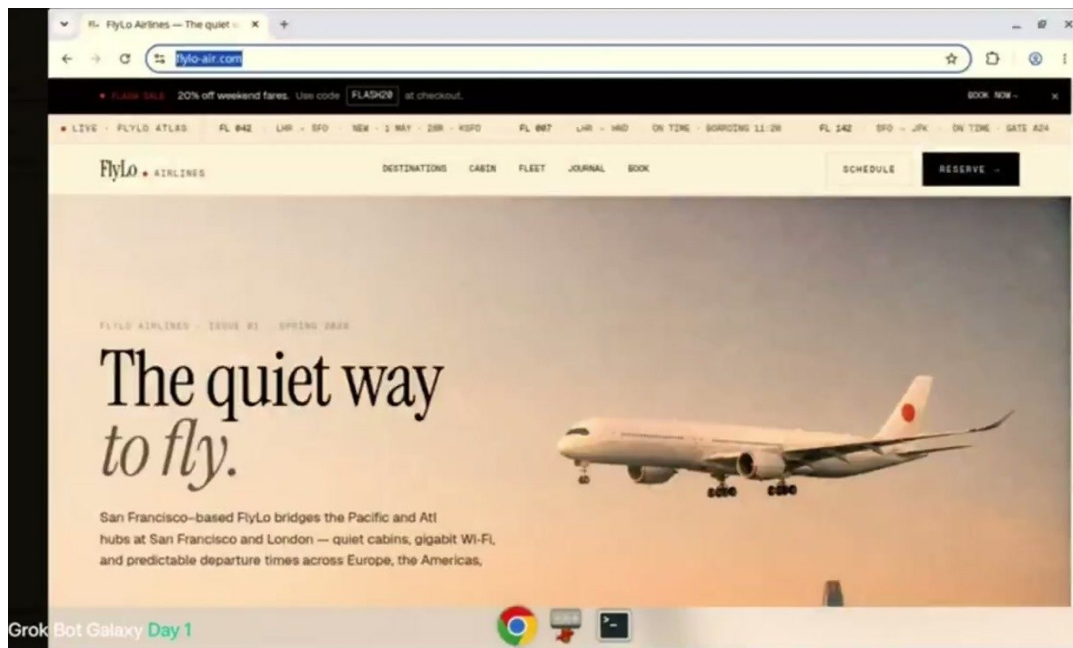
למה שלושה engineer bots ולא אחד, אם הם רצים על אותו מודל? לפי הדובר, כל bot מסוגל לפתור כל משימה. הבעיה היא context: לכל אחד pipeline קצת אחר ומגבלת context משלו. כשמפעלים, מה שנאמר ל-Hogan נשמר בזיכרון של Hogan, ומשימה חדשה מתחילה מה-context הקיים בלי לטעון את ההוראות מחדש. "This can make" [23:22] "your fleet runs much faster". מדברים בעיקר עם ה-chief of staff, והוא מנתב.



מסך הדמו: שיחה ב-Grok Bot שבה Craig, ה-engineer הוותיק, מעביר ל-Nightly Audit Engineer החדש את זרימות העבודה. ברשימה מימין רואים את ה-bots בצוות

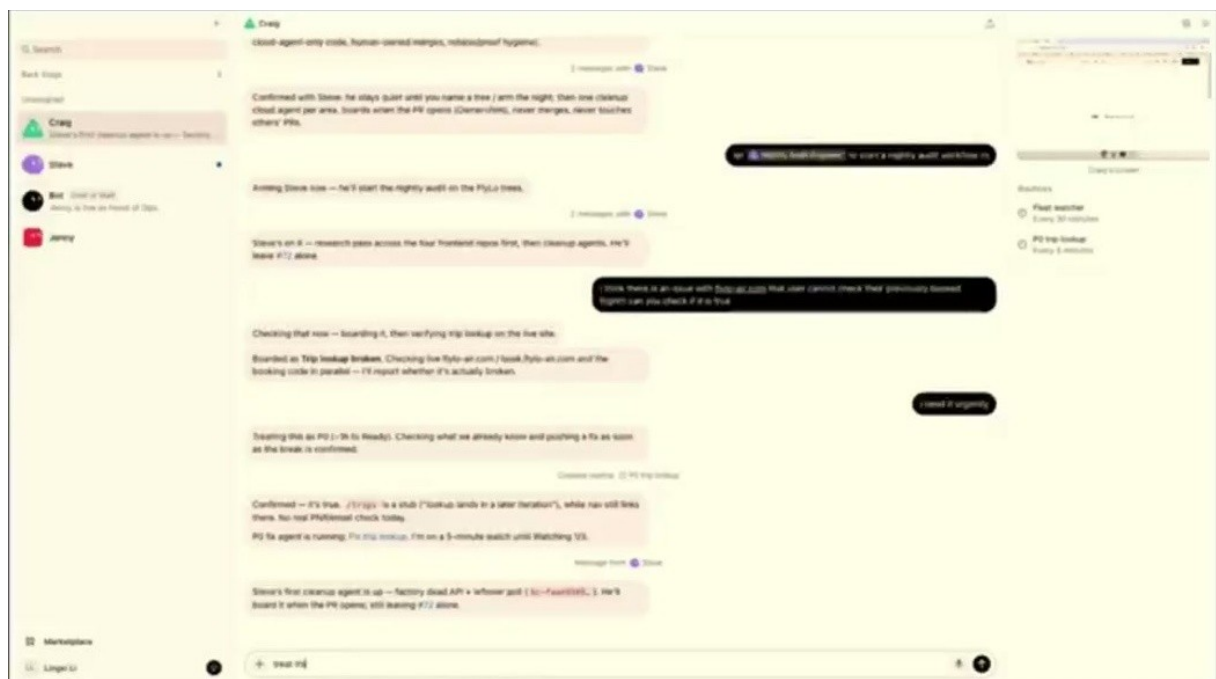
הדובר מכריז ש-"This will be a very live demo because I'm also the first time setting these things". [23:22] הוא מוריד מה-marketplace את ה-nightly audit engineer, וה-bot מתחיל onboarding בעצמו. אבל ל-bot החדש אין את ההקשר של זרימות העבודה הקיימות. במקום להסביר לו הכול, הדובר מבקש מ-Craig לקלוט אותו, ובאותה הזדמנות לשנות את השם שלו ל-Steve. Craig שולח ל-Steve את המבנה של לוח ה-Notion, את ההגדרה של clean ואת שלבי זרימת העבודה, והשניים מאשרים זה לזה. אחר כך Steve מפעיל nightly audit, ליצור ההדגמה [24:53, 26:25].

במקביל מגיע דיווח דחוף: באתר flyloair.com משתמשים לא מצליחים לראות טיסות שהזמינו. Craig מנווט באתר עם computer use, והדובר מציין שזה מכסה את כל מה שאין לו MCP. Craig מאשר שהתקלה אמיתית: /trips הוא stub, ובכל זאת יש אליו קישור בניווט [27:57, 31:03].



אתר ההדגמה Flylo, עם הכותרת *The quiet way to fly*, כפי שה-bot פותח אותו בדפדפן כדי לבדוק את התקלה בטיסות שהוזמנו

Lingxi [27:57] We don't want that we don't want agent to guess things we want deterministic — results



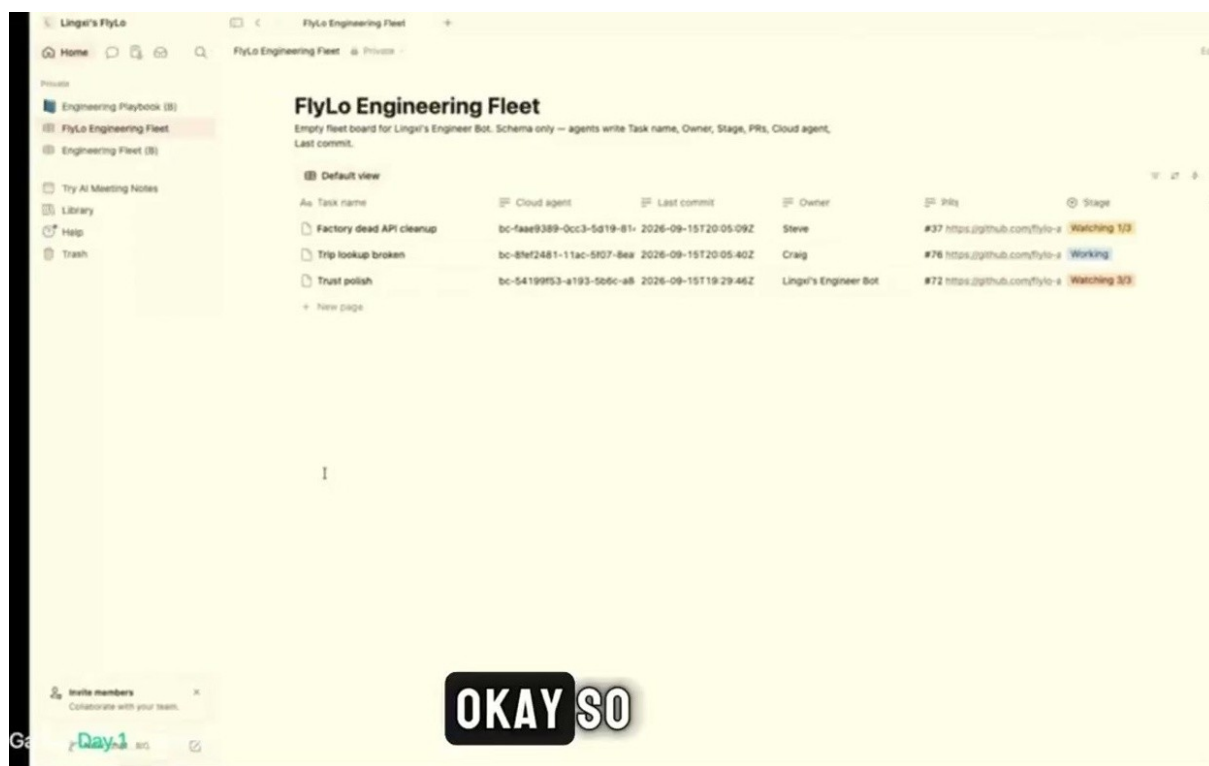
מסך הצ'אט עם Craig שבו הדובר כותב את הגדרת ה-P0: routine שבודק את ה-cloud agents כל חמש דקות, ומה נחשב סטייה מהמסלול

הבעיה לפי הדובר: כשאומרים ל-"agent"urgant, הוא עלול לדלג על שלבים ולנחש. במקום לחזור על זה כל פעם, הוא מגדיר P0 פעם אחת בהודעה רגילה: routine שבודק את ה-cloud agents כל חמש דקות ומזהה סטייה, למשל sleep 300 שמחכה ל-tests שהיו מסתיימים תוך דקה, או זהירות יתר. כשזה קורה ה-bot קוטע את ה-agent ושולח prompt חדש בלי מעורבות של אדם. "The only thing I write is this" [32:37, 34:07]. אחר כך הוא עונה על הודעת התקלה: לתקן, ולטפל בזה כ-P0.

כדי שההגדרה לא תישאר רק אצל Craig, מה ש-"not one scale" עם 10 או 20 bots, ה-chief of staff עושה onboarding ל-Jenny כ-Jenny. head of operations מחזיקה ב-playbook ב-Notion, ואחרים לא מעדכנים אותו. Craig מעביר לה את ה-P0, והיא מודיעה לכל ה-bots ש-"The p0 urgent is now a standing operations".

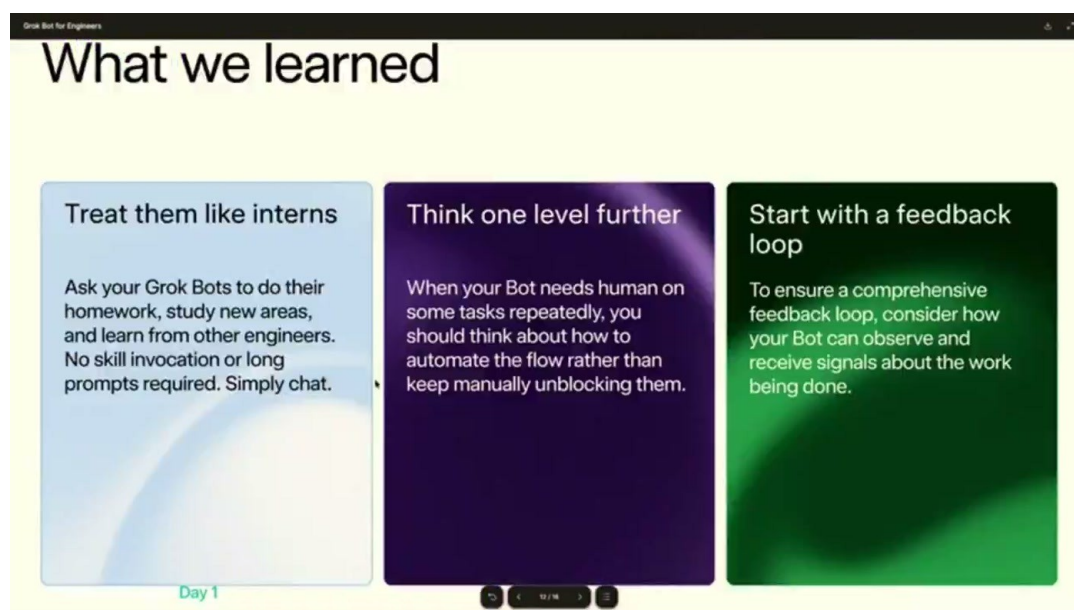
[37:17, 38:51]. מיד אחר כך נוספת דרישה ל-playbook: כל PR מגיע עם הוכחות, screenshots לשינויי UI ו-perf metrics לשיפורי ביצועים.

שני מגנונים שמונעים מה-context להתפוצץ: ה-engineer bot מה-marketplace מגיע עם מסד Notion של ה-fleet, שבו כל משימה והשלב שלה. כך bot שמנהל 20 cloud agents לא צריך לזכור את כולם [40:25]. בנוסף, PRs עוברים Bugbot ו-[40:25] security comments.



מסד Notion בשם FlyLo Engineering Fleet: טבלת משימות עם עמודות לשלב ולבעלים, שה-bots קוראים ממנה במקום להחזיק את המצב ב-context-

6. מה לקחת מכאן



שקופית What we learned עם שלושה כרטיסים: Treat them like interns, Think one level further, Start with a feedback loop

- **Treat them like interns**: לדבר עם ה-bot כמו עם מתמחה מוכשר שלא יודע מה לעשות, בלי להפעיל skills ובלי prompts ארוכים. מסבירים פעם אחת בצ'אט [42:00 בקירוב, בלוק 41:57].

- **Think one level further**: אם אתה חוזר על אותו prompt עשר פעמים, תוציא אותו לשכבה נפרדת: playbook, או bot שמזכיר ל-bot אחר בזמן הנכון [43:27].
- **Start with a feedback loop**: בלי signal ברור של הצלחה מול כישלון, ה-bot לא יודע אם לדחוף הלאה או לאשר merge. הלולאה הפשוטה ביותר היא computer use מול האתר [43:27].
- **צעד מעשי, קריטריונים לפני auto-merge**: להגדיר בכתב מה נחשב הוכחה (screenshot, tests) אמיתיים, perf (before/after) לפני שמאפשרים ל-agent למזג לבד.
- **צעד מעשי, מצב מחוץ ל-context**: כשבודקים fleet, להחזיק את מצב המשימות ב-DB חיצוני ולפצל bots לפי תחום ברגע שה-context הופך לצוואר בקבוק.
- **צעד מעשי, דחיפות כנוהל**: להגדיר P0 כ-routine עם תדירות בדיקה וקריטריוני קטיעה, לא כמילה ב-prompt.
- **צעד מעשי, ללמד להגיד לא**: לכתוב את הגבולות של המוצר ואת הנימוק להם, כדי שה-bot לא יקבל כל feedback כברירת מחדל.

שאלה פתוחה	למה זה חשוב
כמה עולה fleet שרץ 24/7 ומפעיל עשרות cloud agents?	בלי מספר אי אפשר לבדוק את טענת ה-10x מול העלות
אילו מנגנוני הגנה יש על auto-merge של PRs מה-audit הלילי ומה-autofix?	merge אוטומטי של קוד שה-agent עצמו כתב הוא סיכון ישיר ל-production
איך מנוהלות ההרשאות וה-credentials של bots שמחוברים ל-Notion, Slack, Vercel, TestFlight?	bot אחד שנפרץ מקבל גישה לכל השרשרת
מה קורה כש-bot קוטע agent בטעות, או כשה-playbook של Jenny סותר את הזיכרון של bot אחר?	אין הסבר איך מיישבים סתירות בין זיכרונות
האם התקלה ב-Flylo תוקנה, ותוך כמה זמן?	הדמו נעצר לפני התוצאה של המקרה הדחוף
על איזה מודל ה-bots רצים?	נאמר רק שכולם רצים על אותו מודל

7. מילון מונחים

מונח	משמעות
Cloud agent	coding agent שרץ על מכונה בענן של Cursor ולא על המחשב המקומי, ואפשר להריץ כמה כאלה במקביל
Fleet	קבוצה של bots או agents שמנוהלים יחד
Chief of staff	ה-bot שמקבל בקשות מהאדם ומנתב אותן ל-bots המתאימים
Memory / Routines	זיכרון מתמשך לכל bot, ומשימות שמופעלות לפי זמן או trigger
Playbook	מסמך נהלים משותף ב-Notion שכל ה-bots פועלים לפיו
Marketplace	קטלוג של bots מוכנים, כמו nightly audit engineer, שאפשר להוריד ולהתאים
Skill	יכולת או נוהל ארוז שה-agent מפעיל בתוך ה-repository
MCP	פרוטוקול לחיבור agents לכלים חיצוניים, כמו Vercel ו-Slack
Computer use	שליטה של ה-agent בעכבר, במקלדת ובדפדפן של מחשב
Handshake	העברת ידע ואישור הדדי בין שני bots
P0	עדיפות עליונה. כאן זה גם routine שבודק את ה-agents כל חמש דקות וקוטע אותם כשהם סוטים
Slop	קוד מיותר או מסורבל שנוצר בכתיבה מהירה עם agents
Monorepo	repository אחד שמכיל את כל הקוד של הארגון
On-call	המהנדס התורן שמטפל בתקלות
Proof	הוכחה שמצורפת ל-PR, כמו screenshot או perf metrics, שעליה מתבסס ה-review
Feedback loop	signal ברור של הצלחה או כישלון, שממנו ה-agent מבין אם להמשיך או לעצור
Private worker	מכונה פרטית, למשל Mac, שה-cloud agents יכולים לרוץ עליה
Bugbot	כלי שבודק PRs ומצביע על באגים שה-agent יצר