

שיחה עם צוות Claude Code: מ-agent בודד ל-loops, ל-routines, ולא לפי agents

שני אנשים מצוות Claude Code ב-Antropic מסבירים איך הם עובדים היום:
Claude, auto mode, routines, verification, במרכז הארגון ועבודה מהטלפון

שני אנשים מצוות Claude Code ב-Antropic (השמות לא נאמרים בהקלטה) · המקור המקורי לא מזהה (repost ב-X של Codez) · 2026-06-08

משך: כ-19 דקות · X

סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זו שיחה מצולמת של כ-19 דקות בין שני אנשים מצוות Claude ב-Antropic. הם מדברים בגוף ראשון על "כשהשקנו את Claude Code" ועל "הצוות שלנו", אבל לא מציגים את עצמם בשם. הכיתוב של ה-repost ב-X מייחס את הדברים ל-"Claude Code creator". את השם הזה אי אפשר לאמת מתוך ההקלטה, ולכן הוא לא מופיע כאן. גם האירוע או הערוץ המקורי לא מזוהים. השיחה נפתחת במונטאז' קצר של משפטים שחוזרים בהמשך. ב-0:00 שומעים קטעים מ-10:52 ומ-15:30.

הטענה המרכזית: יחידת העבודה של מהנדס עברה בתוך שנה וחצי שתי קפיצות. קודם מקוד ל-agent, ועכשיו מ-agent ל-loop או routine שמפעילים את Claude במקומו. כדי שזה יעבוד צריך שלושה דברים: verification במובן של "האם ה-agent יכול להריץ את הדבר", לולאת לימוד מכל טעות (Claude.md ו-skills), ומודל הרשאות שמאפשר לסמוך על ה-agent ולעבור לדבר הבא, כלומר auto mode. מעבר לזה הם טוענים שתפקידי ה-PM, המעצב והמהנדס מתמזגים, ושארגונים צריכים לשים את Claude במרכז כל תהליך. בדיוק כמו שהיה צריך לעשות עם המחשב האישי.



פריים מהשיחה: שני הדוברים ליד שולחן עבודה. בסרטון אין שקפים, אלא רק צילומי שיחה וכרטיסי כותרת קצרים.

איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, וגם מה לא נאמר.
- 2. **שתי קפיצות:** מקוד ל-agent ל-loop, והעיקרון שכל טעות הופכת ל-CLAUDE.md או ל-skill.
- 3. **Verification:** למה זה לא unit tests, ואיך עובד ה-desktop development skill.
- 4. **Routines: agents** שמאזינים לבאגים, פותחים PR ומתחזקים code review.
- 5. **Auto mode ואבטחה:** למה plan mode ו-permission prompts יצאו מהשימוש, ואיך בנו אמון ב-classifier.
- 6. **Claude במרכז הארגון:** ההקבלה למחשב האישי בשנות ה-90 והתמזגות התפקידים.
- 7. **סביבת העבודה ו-agent view, worktrees, remote control ו-context:** מהטלפון וגישת context minimalism.
- 8. **מה עושים עכשיו:** צעדים מעשיים וטבלת שאלות פתוחות.
- 9. **מילון מונחים:** המונחים באנגלית שמופיעים במסמך.

הערת מקור המסמך הופק אוטומטית מתמליל ההקלטה ומהפריימים שלה. בסרטון אין שקפים, כך שכל התוכן מבוסס על התמליל. ה-ASR משבש שמות ומונחים: Claude מתומלל לא פעם כ-Sonnet 3.5 או Cloud, Sonic 3.5, ו-programmatically כ-for grammatically. במקומות האלה נכתב המונח הנכון, והמקומות שבהם יש ספק מסומנים. הציטוטים מועתקים מילה במילה מהתמליל, כולל השיבושים, ונבדקו מול חותמת הזמן שלהם. ההקלטה לא מבחינה בין שני הדוברים, ולכן הציטוטים מיוחסים ל"אחד הדוברים". לא נוסף שום מידע ממקורות חיצוניים.

1. תקציר מנהלים

זו לא הרצאה עם נתונים. זו שיחה של שני אנשים מבפנים על דפוסי העבודה שלהם עם Claude Code. הערך שלה הוא בתיאור פרקטיקות קונקרטיות: איך נראית לולאת verification, איך routine מתקן באג לפני שבעל הפיצ'ר בכלל ראה אותו, ולמה הם מעדיפים auto mode על פני אישור ידני. מספרים כמעט אין, וחשוב לזכור את זה כשקוראים את הטענות.

- **המהנדס כבר לא מדבר עם קוד, ואפילו לא עם agent:** לפני שנה וחצי עברו מכתובת קוד לשיחה עם agent. עכשיו מדברים עם loop או routine שמפעילים את Claude בעצמם [10:52]. אחד הדוברים מתאר עץ של אלפי agents שמפעילים agents אחרים [0:00].
- **כל טעות של Claude הופכת לכלל קבוע:** לא אומרים ל-Claude לעשות אחרת, אלא מבקשים ממנו לכתוב את הלקח ל-CLAUDE.md או ל-skill. ככה, לדבריהם, הוא יכול "לרוץ לנצח" [1:33].
- **Verification פירושו שה-agent מריץ את המוצר בעצמו:** זה לא lint או unit tests. ה-agent מפעיל את האפליקציה, למשל דרך desktop development skill שמשמש ב-computer use, בודק edge cases, מתקן ובודק שוב [3:08, 1:33].
- **Routines הם השימוש המובהק הראשון ב-Claude Code באופן תכנותי:** מהנדס אחד הריץ routine שאוסף כל באג על voice mode ופותח routine PR. נוסף שלו מתקן באגים שלא קיבלו תגובה תוך חמש שעות [4:39, 6:12].
- **Plan mode יצא מהשימוש לטובת auto mode:** לפי אחד הדוברים, החל מ-Opus 4.6 ובוודאות עם 4.7 המודלים לא צריכים שלב תכנון נפרד [7:44, 6:12].
- **Auto mode בטוח יותר מקריאת permission prompts (זו טענת הדוברים):** כשמאשרים 99% מהבקשות, העין מפסיקה לקרוא. האמון ב-classifier נבנה על אלפי transcripts, על red teamers ועל [7:44, 9:19].
- **Claude במרכז הארגון, כמו המחשב בשנות ה-90:** ב-Anthropic שואלים את Claude ולא אנשים, כבר מה-onboarding, וגם code review ו-security review עוברים דרכו. אצל המחשבים המעבר לקח 10–15 שנה, ועם AI הוא קורה מהר יותר [12:26, 10:52].
- **התפקידים מתמזגים:** PMs, מעצבים, צוות ה-finance וה-data science עובדים ב-Claude Code, ומהנדסים מובילים מוצר מקצה לקצה [13:59, 4:39, 3:08].
- **סביבת העבודה: מבוטלים קודמים לטלפון:** במקום שש לשוניות טרמינל עם שישה agent view: checkouts, app-desktop שמנהל worktrees ו-remote control מהטלפון. לגבי context ההמלצה היא מינימליזם [13:59, 17:00].

מה לא נאמר

- **אין מספרים:** אין נתוני פרודוקטיביות, כמות PRs, שיעור תיקונים שמוזגו או עלויות. על ה-classifier של auto mode נאמר רק שבדקו אותו על "thousands of transcripts", בלי שיעורי false positive או false negative.
- **הכיתוב ב-X לא תואם את התמליל:** "a hundred agents" לא נאמר. נאמר "I have so many agents running", ובמקום אחר "hundreds or thousands". גם "50%" לא נאמר: בתמליל מופיע "probably half my engineering now", והמשך המשפט על הטלפון משובש ב-ASR.
- **Loop לא מוסבר:** לא ברור אם הכוונה לפקודה /loop, כמו שהכיתוב טוען, או לקונספט כללי. אין הסבר איך מגדירים loop או routine.
- **אין מנגנוני בקרה ל-routines:** נאמר רק שהמהנדס ממזג את התיקונים ש-"easy to verify". לא דובר על הרשאות, על מניעת תיקונים שגויים או על rollback.
- **השאלה על context engineering בארגון גדול נענתה באופן כללי:** לא ניתנה הנחיה ספציפית לארגונים גדולים מעבר ל-"minimal system prompt".
- זהות הדוברים והאירוע לא נאמרים בהקלטה.

2. שתי קפיצות: מקוד ל-agent ול-loop

הנקודה ההיסטורית: אחד הדוברים מספר שכשהעלו לראשונה סרטון של Claude Code ל-Slack הפנימי, רק שני אנשים הגיבו. הדוברת השנייה מודה שבהתחלה הוא היה טוב [0:00] "for my very easy engineering tasks".

שנה אחרי ההשקה הם מתארים "armies of agents" ו-agent שמפעיל "a tree of thousands of agents", [0:00] agents.

הקפיצה הראשונה, לפני כשנה וחצי: "I don't write the source code. I talk to an agent" [0:00, 10:52]. הקפיצה השנייה קורית עכשיו, והמהנדס מדבר עם loop או routine. העיקרון שמחזיק את כל זה הוא לולאת לימוד. כש-Claude טועה, התיקון נכנס לזיכרון הקבוע ולא נשאר בשיחה.

— אחד הדוברים [10:52] I don't talk to an agent anymore. I talk to a loop or I talk to a routine. And it prompts Quad for me.

— אחד הדוברים (quad MD הוא CLAUDE.md) [1:33] I don't tell quad to do it differently. I tell it to write it to the quad MD or to like make a skill or something to do it differently.

3. Verification: האם ה-agent יכול להריץ את הדבר

לדבריהם, זה הנושא שכולם מפספסים. כשמדברים על verification אנשים חושבים על unit tests, lint ו-type check, אבל אלה דברים שכבר היו אוטומטיים ממילא. verification ל-agents אומר שה-agent מפעיל את המוצר בפועל, ומציאת הדרך לאפשר לו את זה דורשת עבודה מחשבתית כי היא "often not straightforward" [1:33]. הרגע המכונן היה עם Opus 4: ביקשו מ-Claude לבנות פיצ'ר ולבדוק את עצמו ב-Bash, והוא פתח Claude CLI ובדק את הפיצ'ר של עצמו. היום יש לולאות כאלה ל-iOS simulator, ל-Android simulator ולמחשבי desktop. [1:33]

הדוגמה הקונקרטית היא ה-desktop development skill, שמהנדס בצוות כתב. ה-skill מלמד את Claude להריץ את ה-desktop app מקומית. כשנוסף UX חדש, Claude לוחץ עליו, בודק edge cases, מתקן ובודק שוב. כשסביבת ה-staging נופלת, Claude קורא ב-Slack כדי להבין אם staging למטה או אם מישהו כבר נתקל בבעיה. בסוף הדיבוג מבקשים ממנו לעדכן את ה-skill עצמו. זו אותה לולאת לימוד מהפרק הקודם, הפעם על verification. [3:08]

— אחד הדוברים [1:33] But actually when we talk about verification for agents, it's something slightly different. It's like, can the agent run the thing?

— אחד הדוברים [3:08] What this skill does is Cloud actually spins up a local desktop app and it uses computer use to click around on it.

4. Routines: agents שמגיבים לבאגים לפני שאתה מגיע אליהם

Routines הם ה-use case שהכי מלהיב אותם כרגע. מהנדס שהשיק voice mode בכל המוצרים הגדיר routine שמאזין לכל ticket, GitHub issue, ודיווח באג על voice mode. ה-Claude שלו מכין תיקון ביוזמתו ושולח לו את ה-PR. אחר כך הוסיף routine דומה לשאר הפידבק שלא קיבל מענה [4:39].

האנקדוטה: אחד הדוברים שחרר פיצ'ר עם edge case שלא שם לב אליו, ומישהו פתח עליו באג. עוד לפני שהגיע לטפל בו, ה-Claude שלו דיווח ש Claude אחר כבר תיקן אותו. התברר שלאותו מהנדס יש routine שמחפש באגים בלי תגובה מעל חמש שעות [6:12]. ההערה הרחבה יותר: ה-Agent SDK היה הניסיון הראשון להשתמש ב-Claude Code באופן תכנותי (ב-ASR: for grammatically), אבל בהתחלה לא היה ברור למה routines הם "the first really obvious application". הם מבצעים את כל ה-code review ו-"babysits every PR": מגיבים להערות, מתקנים CI ועושים rebase. לדברי הדובר, אלה דברים שהוא "haven't done in a long time" [6:12].

— אחד הדוברים [6:12] he has another routine that just looks for bug reports that haven't been responded to in five hours and puts up a fix and he merges the ones that are easy to verify.

5. Auto mode: למה הוא החליף את plan mode ואת אישורי ההרשאות

בשאלה מה הפיצ'ר המרכזי בעבודה סינכרונית ב-CLI, התשובה מפתיעה: פעם זה היה plan mode, והיום זה auto mode. הנימוק הוא שהתכנון היה קריטי בערך מ-Opus 4 עד 4.5, וממודל 4.6 ובוודאות ב-4.7 הוא כבר לא נחוץ. יש מי שעדיין אוהב את ה-artifact של התכנון [6:12, 7:44]. היתרון המעשי: מפעילים Claude, עוברים ל-Claude הבא ולא צריך לשבת ולצפות.

מודל ה-permission prompts המקורי, שבו כל tool דרש כן או לא, היה "the best we had" לפני שנה וחצי, כשלא היו classifiers והמודל היה פחות aligned. ב-auto mode הבקשה מנותבת למודל נפרד שבודק אבטחה. אם משהו נראה "a little sus", הוא נחסם, ואפשר לאשר אותו ידנית אחר כך [7:44]. אחד הדוברים מודה שכשהציעו את זה לראשונה הוא חשב שזה לא יעבוד ("route the prompt to a model no way"), אבל אמפירית זה עבד. לדבריו, הרבה מהידע ההנדסי הקודם פשוט צריך לזרוק [9:19, 10:52].

— אחד הדוברים [7:44] It's just human nature when you accept 99% of requests that your eyes just glaze over when you read it. And so actually we feel that auto mode is more safe than reading every single permission prompt because it means that you're only paying attention to the most important thing

- **שלב 1, classification:** אספו אלפי transcripts של trajectories מלאים של agent יחד עם permission prompt, וביקשו מ-auto mode לסווג אם הבקשה בטוחה. לדבריהם הוא היה "extremely good" [9:19].
- **שלב 2, red teamers חיצוניים:** ביקשו מהם לנסות prompt injection ולפרוץ את ה-codebase, ומהניסיונות בנו evals כדי לוודא שכולם נחסמים [9:19].
- **שלב 3, תקיפה פנימית:** צוותים פנימיים ניסו לפרוץ את auto mode, והוא שופר עד שתפס את כל הניסיונות. המטרה היא הגנה גם מפני "the most intelligent attacks that we can construct" [9:19].
- **מה חסר:** לא נמסרו שיעורי טעות, וגם לא נאמר מה קורה כשה-classifier מאשר פעולה מזיקה.

6. Claude במרכז הארגון, והתמזגות התפקידים

אחד הדוברים מצטט case study מ-Harvard Business Review משנות ה-90, שעסק בשאלה למה המחשבים לא מביאים את הפרודוקטיביות שציפו לה. התשובה הייתה שלא מספיק להציב מחשב ליד ארון התיוק. צריך לזרוק את ארון התיוק, את הנייר ואת העטים, ולשים את המחשב במרכז כל תהליך עסקי [10:52, 12:26]. ב-Anthropic, לדבריו, מי שמצטרף לא שואל אנשים אלא את Claude. code review, security review, ומילוי טפסים עוברים דרך Claude ו-Cowork (ב-ASR: co-work). מאחר שרוב העבודה כבר דיגיטלית ו-Claude יודע להפעיל מחשב, המעבר מהיר יותר מ-10–15 השנים שלקח המעבר למחשבים [12:26].

במקביל התפקידים מתמזגים. ה-PM כותב קוד, ומעצבת (השם במקור, Megan, לפי ה-ASR) מעלה PRs שמתקנים כפתור [3:08]. מעצבים בונים prototypes ישירות באפליקציה במקום לבקש ממהנדס, צוות ה-finance מריץ את התחזיות שלו ב-Claude Code, וגם ה-data scientists עובדים בו [4:39]. לשאלה אם כולם יהיו PMs או מהנדסים, התשובה היא שניהם. מהנדסים מובילים פיצ'ר מקצה לקצה, כולל עבודה מול legal, marketing ו-security. לדבריהם, AI מתגמל סקרנות, טעם מוצרי ו-end-to-end ownership [13:59]. אחד הדוברים מוסיף שאין לו כבר to-do list, כי Claude בונה כל רעיון [13:59].

— אחד הדוברים [13:59] Everyone's going to be both. I feel pretty strongly that these roles are merging

7. סביבת העבודה: agent view, worktrees, הטלפון ו-context מינימלי

עד לא מזמן, אחד הדוברים עבד עם שש לשוניות טרמינל ושישה Git checkouts של אותו repo, ועבר ביניהן. היום יש לו לשונית אחת עם ה-agent view החדש, וה-desktop app מנהל בשבילו את ה-worktrees [13:59, 15:30]. ההפתעה מבחינתו היא שבערך חצי מהעבודה ההנדסית שלו נעשית היום מהטלפון (ההמשך משובש ב-ASR: "don't have my phone"). הוא מתחיל משימה במחשב, מתחבר אליה ב-remote control, הולך לקנות קפה, בודק

מה קורה עם ה-agents, ולפעמים מפעיל agent חדש ב-voice mode באמצע שיחה. הדוברת השנייה מספרת שראתה את המחשב שלו נשאר נעול על השולחן במשרד ערב אחרי ערב, בזמן שה-PRs שלו המשיכו להתמזג, "m'l coding from my couch" [15:30].

לגבי context engineering, הם מתארים התפתחות לפי מודלים. בתקופת Sonnet 3.5 (ב-ASR: Sonic 3.5) עשו prompt engineering, בתקופת Opus 4 עשו context engineering, והיום נותנים system prompt ו-tools מינימליים ודרך למשוך context לפי הצורך [17:00, 15:30]. הדוברת השנייה מגדירה את עצמה context minimalist: יותר מדי context הוא micromanagement, והמודל לפעמים מוצא דרך טובה יותר. גם ה-harness של Claude Code נעשה רזה יותר כדי להשאיר מקום לפרומפטים של המשתמש [17:00]. לסיום הם צופים שבעוד שנה צורת העבודה תהיה שונה לגמרי, כי agents רצים יותר זמן, עצמאיים יותר ופועלים בכמויות גדולות. לדבריהם הרעיונות יגיעו מהצוות ומהקהילה ולא מהם [18:31, 17:00].

— אחד הדוברים [17:00], You give it the minimal possible system prompt, the minimal possible tools, and then you let the model figure it out.

8. מה עושים עכשיו

- הפכו כל תיקון לכלל קבוע: כשה-agent טועה, אל תסתפקו בתיקון בשיחה. בקשו ממנו לעדכן את CLAUDE.md או skill רלוונטי.
- בנו verification שמריץ את המוצר: skill שמסביר ל-agent איך להרים את האפליקציה מקומית ולהפעיל אותה (computer use, simulator), ואיך לעדכן את ה-skill כשמשהו נשבר.
- התחילו מ-routine אחד צר: למשל באגים על פיצ'ר אחד, עם PR שמגיע לבן אדם. מזגו רק תיקונים שקל לאמת, כמו שתואר בשיחה.
- בחנו auto mode מול אישור ידני: הטענה שהוא בטוח יותר מגיעה מהספק ולא נתמכת בנתונים. כדאי לבדוק אותה מול מודל האימונים של הארגון לפני אימוץ.
- בדקו מחדש את ההרגל של plan mode: עם מודלים חדשים ייתכן שהוא מיותר. השאירו אותו אם ה-artifact של התכנון חשוב לכם.
- קצצו CLAUDE.md: context: וכלים מינימליים, ודרך למשוך context לפי הצורך במקום לדחוף את הכול מראש.
- עבודה מקבילית: worktrees במקום checkouts ידניים, ו-remote control כדי לעקוב אחרי agents גם הרחק מהמחשב.

שאלה פתוחה	למה זה חשוב
מה שיעורי ה-false positive וה-false negative של ה-classifier ב-auto mode?	ההחלטה אם לוותר על אישור ידני תלויה בזה, ולא נמסר שום נתון.
איך מגדירים loop ו-routine בפועל, ואיפה הם רצים?	השיחה מתארת תוצאות בלי מנגנון, כך שאי אפשר לשחזר אותן מהסרטון.
מה מונע מ-routine למזג תיקון שגוי?	נאמר רק שממזגים תיקונים "easy to verify". אין פרטים על בקרה או rollback.
מה העלות של מאות או אלפי agents?	לא דובר על tokens, עלויות או מגבלות קצב.
איך עושים context engineering בארגון גדול עם הרבה repos ותלויות?	זו הייתה שאלה ישירה, והתשובה הייתה כללית.
האם הטענה על Opus 4.6 ו-4.7 בלי צורך בתכנון נכונה גם במשימות ארוכות ומורכבות?	מבוסס על חוויה אישית בלבד, בלי השוואה.

9. מילון מונחים

מונח	משמעות
agent	מודל שפועל עצמאית עם כלים, מריץ פקודות וכותב קוד
CLAUDE.md	קובץ הוראות קבוע ש-Claude Code קורא בכל הפעלה, לזיכרון כללים ולקחים
skill	חבילת הוראות ויכולות שמלמדת את Claude לבצע משימה מסוימת, למשל להריץ את ה-desktop app

מונח	משמעות
verification	אימות שהקוד עובד. בשיחה: ה-agent מריץ ומפעיל את המוצר בעצמו
computer use	יכולת של המודל להפעיל ממשק גרפי: ללחוץ, להקליד ולצלם מסך
routine	תהליך מתוזמן או מבוסס אירועים שמפעיל את Claude בלי התערבות אדם
Loop	לולאה שמפעילה את Claude שוב ושוב. לפי הכיתוב ב-X-הכוונה לפקודה /loop, וזה לא מאומת בתמליל
Agent SDK	ערכת פיתוח להפעלת Claude Code באופן תכנותי
plan mode	מצב שבו Claude מתכנן ומציג תוכנית לפני ביצוע
auto mode	מצב שבו classifier נפרד מאשר או חוסם פעולות במקום לשאול את המשתמש
permission prompt	בקשת אישור מהמשתמש לפני הרצת tool
classifier	מודל שמסווג בקשה כבטוחה או לא בטוחה
red teaming	תקיפה מכוונת של מערכת כדי למצוא חולשות
prompt injection	הזרקת הוראות זדוניות לתוכן שה-agent קורא
evals	מבחנים אוטומטיים שמודדים התנהגות של מודל
worktree	עותק עבודה נוסף של אותו Git repo, לעבודה מקבילית
agent view	תצוגה ב-Claude Code לניהול כמה agents במקביל
remote control	שליטה מרחוק, למשל מהטלפון, ב-session שרץ במחשב
voice mode	הפעלת Claude בדיבור
context engineering	בחירה מכוונת של המידע שמוזן למודל
harness	שכבת התוכנה שעוטפת את המודל: system prompt, tools ולולאת הריצה
Cowork	כלי של Anthropic שמוזכר כמי שממלא טפסים (השם מבוסס על ה-ASR)