

# Claude Code: שיטות עבודה מומלצות

מהנדס מצוות Claude Code ב-Antropic מסביר איך הכלי בנוי, ומזה גוזר איך כדאי לעבוד איתו: CLAUDE.md, הרשאות, ניהול context, תכנון לפני קוד ועבודה עם כמה Claude במקביל

Cal Rueb (Anthropic) · מקור מקורי: Code w/ Claude, לפי שקופית הפתיחה. הסרטון הועלה מחדש ב-X על ידי darkzodchi · 29.4.2026 (תאריך ההעלאה המחודשת)

משך: כ-26 דקות · X

סיכום אוטומטי — (aibriefing.dev) AI Briefing

## על המסמך הזה

זו הרצאה של כ-26 דקות בשם "Claude Code best practices". המרצה הוא Cal Rueb, שהגיע ל-Anthropic לפני כשנה וחצי כדי להקים את צוות Applied AI. כיום הוא core contributor בצוות Claude Code ועובד על system prompts, על תיאורי הכלים ועל ה-evals של המוצר. לפי שקופית הפתיחה ההרצאה ניתנה באירוע Claude w/ Claude. הכיתוב של מי שהעלה אותה מחדש ב-X מדבר על "workflow מלא" ועל "template", אבל אף אחד מהם לא מופיע בהרצאה עצמה.

הטענה המרכזית: Claude Code הוא agent "טהור". יש לו הוראות, כלים חזקים ולולאה שרצה עד שהמודל מחליט שסיים. אין בו אינדוקס ואין RAG, רק agentic search עם הכלים שמפתח היה משתמש בהם בטרמינל. מי שמבין את זה מבין גם את שיטות העבודה: לתת לו זיכרון דרך CLAUDE.md, לכוון הרשאות, לנהל את ה-context window, לבקש תוכנית לפני קוד, להשתמש ב-Escape בזמן הנכון ולהריץ כמה מופעים במקביל.



שקופית הפתיחה: Claude Code best practices, הולוגו Code w/ Claude ושם המרצה Cal Rueb, Member of Technical Staff, Anthropic

### איך לקרוא את המסמך

- 1. **תקציר מנהלים:** הטענות המרכזיות בעמוד אחד, ומה לא נאמר.
- 2. **איך Claude Code עובד:** מודל המנטלי, ה-agent loop, agent מול אינדוקס, הרשאות ואבטחה.
- 3. **Use cases:** חמשת השלבים של מחזור הפיתוח, מ-onboarding ועד מיגרציות גדולות.
- 4. **סביבה וקונפיגורציה:** CLAUDE.md, ניהול הרשאות, אינטגרציות (MCP מול CLI), clear/ ו-compact.
- 5. **Workflows וטכניקות מתקדמות:** תוכנית ו-vibe coding, TODOs עם guardrails, צילומי מסך, כמה Claude במקביל, MCP, Escape, headless.
- 6. **דמו ושאלות מהקהל:** thinking/ model, בין קריאות לכלים, אינטגרציה עם IDE, ותשובות על CLAUDE.md ועל multi-agent.
- 7-8. **פעולות ומילון:** מה לעשות מחר בבוקר, שאלות פתוחות ומילון מונחים.

**הערות מקור** המסמך הופק אוטומטית מהתמלול ומהשקופיות של ההקלטה. התמלול נוצר ב-ASR, ולכן שמות ומונחים עלולים להשתבש. למשל, Claude Code מופיע בתמלול כ-"Cloud Code", והשמות Boris, Kat ו-Koo נרשמו כפי שנשמעו. הציטוטים מועתקים מילה במילה מהתמלול ובבדקו מול חותמת הזמן שלהם, ולכן כתיב ה-ASR נשאר בהם. לא נוסף מידע ממקורות חיצוניים.

# 1. תקציר מנהלים

זו לא הרצאת מוצר שיווקית. מהנדס שכותב את ה-prompts של הכלי מסביר מה קורה מתחת למכסה המנוע, ומשם גוזר המלצות עבודה. רוב ההמלצות פשוטות, אבל כמה מהן חשובות במיוחד: להעדיף CLI על פני MCP, להשתמש ב-compact/, להקיש Escape פעמיים, ולשתף state בין agents דרך קובץ.

- **אין אינדוקס, יש Claude Code agentic search:** לא מטמיע (embed) את הקוד ולא מריץ RAG. הוא מחפש עם Find ו-Glob, Grep כמו מפתח חדש בצוות, וחוזר לחפש לפי מה שמצא. "We don't do any sort of indexing." [4:34]
- **ה-agent הוא לולאה פשוטה:** הוראות, כלים חזקים, והמודל רץ בלולאה עד שהוא מחליט שסיים. זו הגישה ש-Anthropic מכנה "the simple thing that works" [3:02].
- **CLAUDE.md הוא הזיכרון היחיד:** בתחילת כל session הקובץ שבתיקיית העבודה נטען ל-prompt. אפשר לשים אותו בפרויקט (ב-git) או בתיקיית הבית [10:41].
- **CLI עדיף על MCP כשיש ברירה:** כשיש CLI מוכר ומתועד היטב, כמו Rueb, gh ממליץ להשתמש בו במקום ב-MCP server [12:13].
- **ה-context נגמר, וצריך לנהל אותו:** חלון של 200,000 טוקנים. clear/ מתחיל מחדש ושומר רק את compact. /compact CLAUDE.md מסכם את ה-session כ-handoff למפתח אחר [13:44].
- **קודם תוכנית, אחר כך קוד:** מבקשים שתיים-שלוש אפשרויות בלי לכתוב קבצים, בודקים את ה-TODO list, ועוצרים עם Escape כשהכיוון לא נכון [7:36], [15:15].
- **Vibe coding עם TDD guardrails:** שינויים קטנים, בדיקות TypeScript ו-lint, ו-commits תכופים שאפשר לחזור אליהם [15:15].
- **כמה Claude במקביל:** יש באנתרופיק מי שמריצים ארבעה במקביל. Rueb עצמו מריץ שניים [16:46].
- **Claude 4 חושב בין קריאות לכלים:** מפעיל "think hard" גם בין קריאות לכלים, וגם CLAUDE.md נאסף טוב יותר. לכן כדאי לעבור שוב על הקובץ [18:17], [22:49].

## מה לא נאמר

- **אין מספרים:** אין נתוני תפוקה, עלות או טוקנים. גם "כיסוי בדיקות גבוה באופן חריג" בקוד של הצוות נאמר בלי מספר [7:36].
- **אבטחה בקצרה בלבד:** השקופית מזכירה Security, אבל בפועל נאמר רק שאפשר להפנות את הכלי למודלים דרך AWS ו-GCP. לא דובר על sandboxing ולא על סיכונים של auto-accept.
- **Multi-agent אין מנגנון מובנה:** בשאלה על שיתוף context בין agents נאמר במפורש שהפתרון הטוב ביותר כרגע הוא קובץ markdown משותף [24:22].
- **Headless/SDK בלי דוגמה:** "the thing we're most excited about", אבל מעבר ל-GitHub Actions לא הוצג שום שימוש קונקרטי [18:17].
- **אין דיון בכשלים:** מלבד בעיית ההערות המיותרות של 3.7 לא הוצגו מקרים שבהם הכלי נכשל או מתי לא כדאי להשתמש בו.

## 2. איך Claude Code עובד

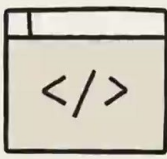
**Claude Code is like that  
coworker that does  
everything in the terminal  
and never touches a GUI**

המודל המנטלי של המרצה: Claude Code הוא עמית שעושה הכול בטרמינל ולא נוגע ב-GUI

Rueb משווה את הכלי למנטור שהיה לו כמהנדס צעיר: מישוהו שפותח טרמינל, מריץ פקודות bash ועורך ב-Vim בלי לגעת בממשק גרפי. מזה נגזר העיצוב. Anthropic מגדירה agent כ-"some instructions, some powerful tools, and you let the model just run in a loop until it decides it's done" [3:02]. הכלים הם בדיוק מה שמשתמש טרמינל חזק היה רוצה: יצירה ועריכה של קבצים, הרצת פקודות, ובנוסף משיכת יכולות חיצוניות דרך MCP.

את ההיכרות עם ה-codebase הכלי בונה ב-agentic search ולא בגישה ש"הייתם בוחרים לפני שנה", כלומר אינדוקס, embeddings ו-RAG. המודל מחפש, קורא את התוצאות, מחליט שחסר לו מידע ומחפש שוב. ההמלצות בהמשך נובעות מזה: בגלל שאין אינדוקס, כדאי לתת לו מפה של הפרויקט ב-CLAUDE.md.

### How Claude Code works

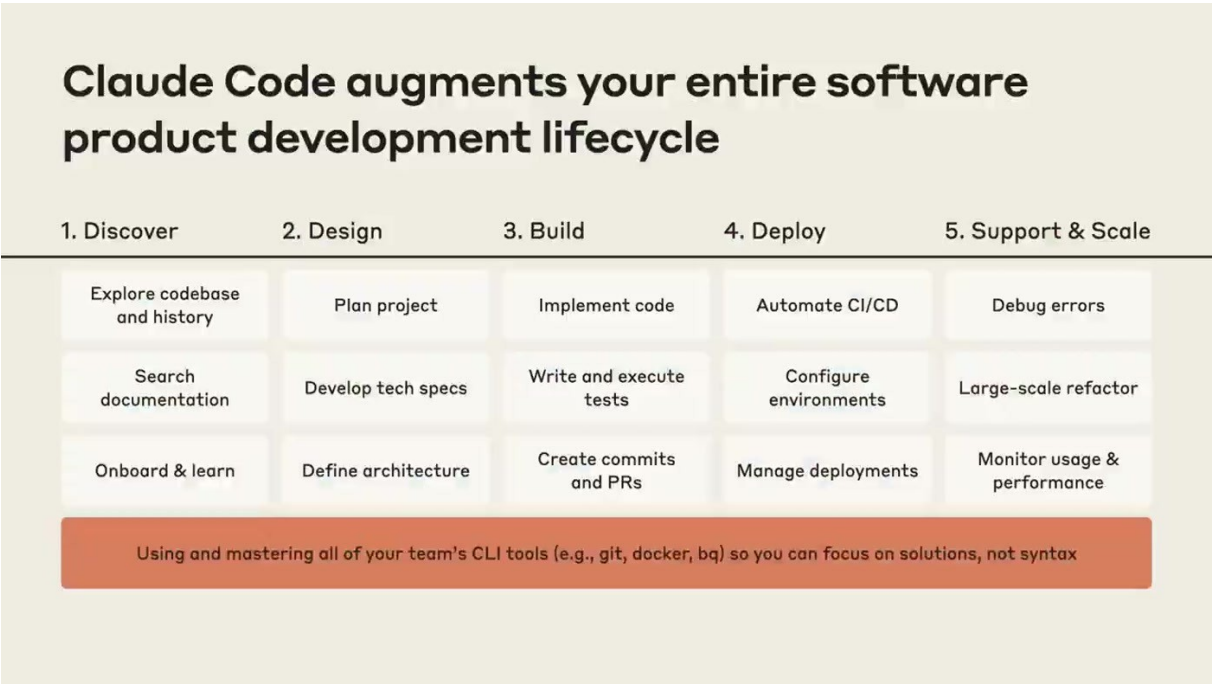


|                                 |                                                                                                                                             |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Powerful actions & integrations | Edits & creates files, uses your CLI and MCP tools, and creates commits. Tackles both narrow tasks like bug fixes and broad refactors       |
| Codebase awareness              | Maintains awareness of your project structure. No need to manually add files to context—Claude agentically searches your codebase as needed |
| Transparency                    | Claude shows its work, and uses a tiered permissions system that balances power with control                                                |

שקופיות How Claude Code works: יכולות ואינטגרציות (קבצים, CLI, MCP, commits), Codebase awareness (חיפוש agentic בלי להוסיף קבצים ידנית) ו-Transparency (מערכת הרשאות מדורגת)

מעל ה-agent יש שתי שכבות. הראשונה היא UI קל שמציג את העבודה בזמן אמת. השנייה היא מערכת הרשאות, שמאפשרת ל-agent לעבוד לבד ומחייבת את האדם להתערב כשהפעולה מסוכנת. השכבה הרביעית בשקופיות היא Security, וכאן ההסבר קצר: Claude Code הוא שכבה דקה מעל המודל, והמודל זמין גם דרך AWS ו-GCP. לכן אפשר להפנות את הכלי לספקים האלה אם זה מתאים לארגון יותר.

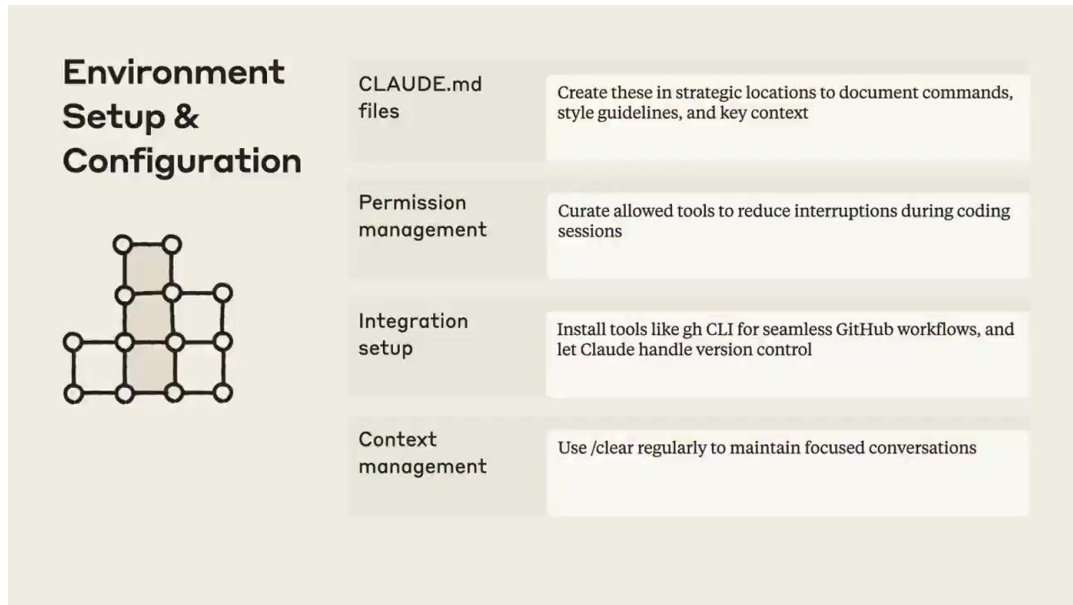
### 3. Use cases לאורך מחזור הפיתוח



חמשת השלבים של מחזור הפיתוח (Discover, Design, Build, Deploy, Support & Scale) והמשימות בכל אחד. בפס האדום: שליטה בכל כלי ה-CLI של הצוות (git, docker, bq) כדי להתמקד בפתרון ולא בסינטקס

- Discover: כשמגיעים ל-codebase חדש אפשר לשאול איפה פיצ'ר ממומש, או לבקש מהכלי לקרוא את ה-git history של קובץ ולספר איך הקוד השתנה בשבועות האחרונים.
- Design: לדברי Rueb זה השימוש שהכי פחות מעריכים, Claude כשותף לחשיבה. הוא מבקש לחקור את הקוד ולחזור עם שתיים או שלוש אפשרויות, ומוסיף במפורש "Don't start working. Don't start writing any files yet." [7:36].
- Build: הדמו של "בנה לי אפליקציה מאפס" מרשים, אבל הצוות מתמקד בעבודה בקוד קיים. יש לזה שתי תוצאות בקוד של הצוות עצמו: כיסוי בדיקות גבוה באופן חריג, והודעות commit ו-PR טובות, כי פשוט מבקשים מ-Claude לכתוב אותן.
- Deploy: שימוש ב-SDK בצורה headless כדי "לפזר" coding agent בתוך CI/CD ו-GitHub.
- Support & Scale: debug מהיר יותר, ומיגרציות גדולות שלקוחות דחו שוב ושוב. Rueb נותן דוגמאות של Java ישנה לגרסה חדשה ושל PHP ל-React או Angular. פרויקט כזה "קל יותר לעיכול" כשיש כלי כזה. הוא מוסיף שבזכות הכלי הוא כבר לא נבהל מ-rebase שנתקע.

## 4. Best practices: סביבה וקונפיגורציה



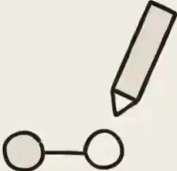
Environment Setup & Configuration: קובצי CLAUDE.md במקומות אסטרטגיים, אוצרות של כלים מורשים, התקנת gh CLI ושימוש קבוע ב-clear

- **CLAUDE.md:** ל-agent אין זיכרון. הקובץ בתיקיית העבודה "נשתל" ב-prompt בתחילת כל session, עם מסר בנוסח "אלה הוראות חשובות שהמפתח השאיר לך". קובץ ברמת הפרויקט נכנס ל-git ומשותף לצוות. קובץ בתיקיית הבית חל על כל העבודה. מה כדאי לכתוב בו: איך מריצים בדיקות, מפת הפרויקט (איפה הבדיקות ומה המודולים), style guide. בונים אותו בהדרגה [10:41].
- **ניהול הרשאות:** קריאה וחיפוש רצים בלי אישור. כתיבה ופקודות bash מקפצות בקשת אישור. Shift+Tab מפעיל auto-accept mode, ובהגדרות אפשר לאשר פקודות ספציפיות קבוע, למשל [12:13] npm run test.
- **אינטגרציות:** כל כלי עם CLI מגדיל את מה שאפשר להעביר ל-Claude, ו-gh הוא הדוגמה. בין CLI מוכר ומתועד לבין MCP server, Rueb ממליץ על ה-CLI. כלים פנימיים מתעדים ב-CLAUDE.md. הוא מזכיר כלי פנימי באנתרופיק שנשמע כמו "Koo", והשם לא ודאי [13:44].
- **ניהול context:** החלון הוא 200,000 טוקנים, ומתקרבים לסופו מקבלים אזהרה. /clear מוחק הכול חוץ מ-CLAUDE.md. /compact מוסיף הודעת משתמש שמבקשת לסכם את העבודה כ-handoff, והסיכום הופך לנקודת הפתיחה של ה-session הבא. הצוות השקיע הרבה בכיוון של הפונקציה הזו.

— Cal Rueb, על מה ש-compact מבקש מהמודל [13:44] I need to go summarize everything we've been up to. I'm going to give this to another developer, and they're going to pick up where I left off

## 5. Workflows יעילים וטכניקות מתקדמות

### Effective workflows



|                                  |                                                                                                             |
|----------------------------------|-------------------------------------------------------------------------------------------------------------|
| Planning and TODOs               | First ask Claude to read relevant files and make a detailed plan. Verify Claude's TODO list.                |
| Smart vibe coding                | Leverage test driven development and regular commits to vibe code with the right level of guardrails        |
| Use screenshots to guide & debug | Use screenshots to compare implementations against mockups, and provide guidance to Claude on how to design |

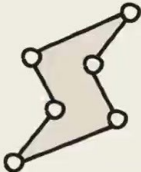
*Effective workflows: תוכנית מפורטת ו- TODO list לבדיקה, vibe coding עם TDD ו-commits תכופים, וצילומי מסך להשוואה מול mockups*

תכנון ו-TODOs: במקום "תתקן את הבאג" מבקשים "תמצא מה גורם לו ותגיד לי מה התוכנית". קוראים את התוכנית ומאשרים לפני שנוגעים בקוד. במשימות גדולות Claude בונה TODO list. אם מופיע בה משהו מוזר, מקישים Escape ומבקשים לשנות אותה.

Smart vibe coding: מותר לתת לו לרוץ, אבל עם guardrails. בדיקות קודם (TDD), שינויים קטנים, הרצת בדיקות, בדיקות TypeScript ו-lint בכל צעד, ו-commit תכופ כדי שתמיד תהיה נקודה לחזור אליה.

צילומי מסך: המודל multimodal. אפשר להדביק צילום מסך או להפנות לקובץ, למשל "look at this mock.png", ולבקש לבנות לפיו.

### Advanced Techniques



|                                |                                                                                                                                                  |
|--------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Multi-Claude & parallelization | Use separate instances for coding, reviewing, and testing. Deploy worktrees or sub-agents to tackle complex problems simultaneously              |
| Use escape                     | Balance letting Claude work and interjecting at the right times. Press Escape once to stop. Press Escape twice to jump back in the conversation. |
| Tool expansion & MCP           | Integrate bash tools, MCP servers, and custom slash commands. Note: Claude Code is both an MCP client & server!                                  |

*Advanced Techniques: מופעים נפרדים לכתיבה, ל-review ולבדיקות (worktrees או sub-agents), Escape פעם אחת לעצירה ופעמיים לחזרה בשיחה, והרחבה עם MCP tools, bash ו-slash commands מותאמים. לפי השקופית, Claude Code הוא גם MCP client וגם MCP server*

- **כמה Claude במקביל:** ב-tmux או בטאבים נפרדים. "I can only do two, but I know people that do four." [16:46]. השקופית מוסיפה פרט שלא נאמר בעל פה: להפריד מופעים לכתיבה, ל-review ולבדיקות, ולהשתמש ב-worktrees.
- [16:46] **Escape:** "escape is your best friend". המיומנות היא לדעת מתי לעצור ומתי לתת לו להסתדר לבד. Escape כפול קופץ אחורה בשיחה, פיצ'ר ש"לא הרבה מכירים".
- **הרחבת כלים ו-MCP:** כשה-bash והכלים המובנים לא מספיקים, זה הזמן לחפש MCP servers.
- **Headless automation:** שימוש תכנותי ב-Claude, כבר קיים ב-Rueb. GitHub Actions מודה שגם באנתרופיק עדיין מנסים להבין איפה עוד אפשר להשתמש בזה [18:17].

## 6. דמו ושאלות מהקהל

```
> /config
  | Set model to Default (claude-sonnet-4-20250514)

> can you figure out what is in this project. think hard

* Thinking...

The user is asking me to figure out what's in this project and to "think hard". Let me start by examining the directory structure that was provided and then use the available tools to explore the project more thoroughly.

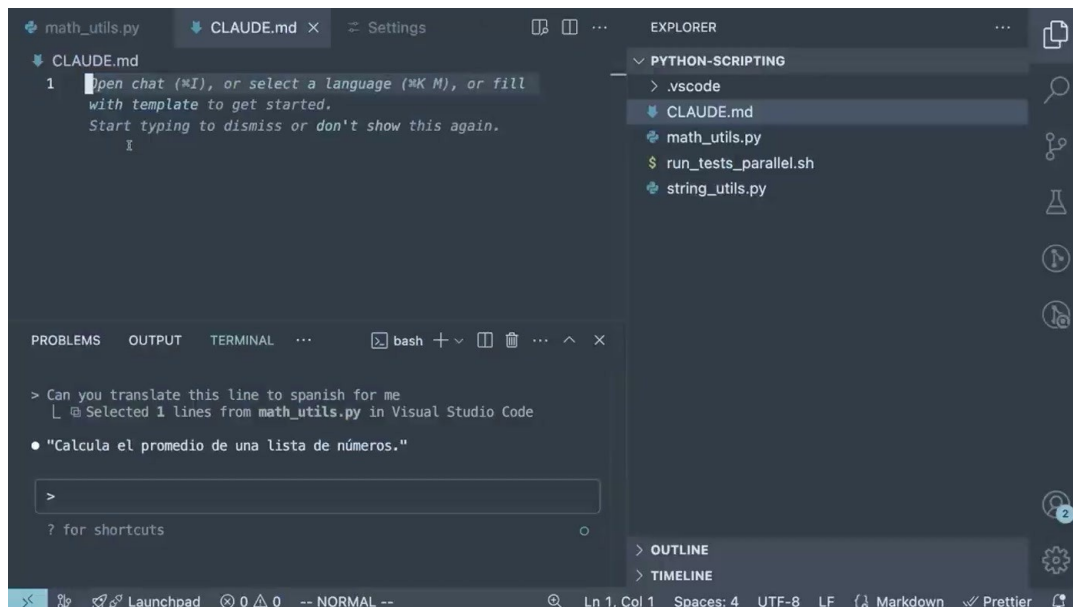
From the directory structure provided, I can see:
- /Users/cal/code/cc-playground/
  - claudemd-package/ (contains CLAUDE.md and package.json)
  - python-scripting/ (contains math_utils.py, run_tests_parallel.sh, string_utils.py)
  - python/ (contains run_tests.sh)

This looks like a playground or testing repository with different components. Let me use tools to explore each directory and understand what's actually in this project.

I should start by reading the key files to understand what this project is about:
1. Read the CLAUDE.md file in claudemd-package/
2. Read the package.json in claudemd-package/
3. Examine the Python files to understand what they do
```

דמו בטרמינל: /config מגדיר את מודל ברירת המחדל (claude-sonnet-4-20250514), הבקשה "think hard" ובלוק ה-Thinking שבו המודל מתכנן לקרוא את CLAUDE.md, את package.json ואת קובצי ה-Python

בדמו Rueb מציג כמה דברים ש"חדשים נכון להיום". /model ו-/config מאפשרים לעבור מ-Sonnet, מודל ברירת המחדל, ל-Opus. ה-thinking השתנה: במודלים הקודמים המודל לא חשב בין קריאות לכלים, "And that's", [18:17] "probably when the thinking matters most." מ-Claude 4 הוא כן חושב שם, ובדמו רואים טקסט thinking באפור בהיר לסירוגין עם קריאות Read. ההמלצה שלו היא להוסיף "think hard" כשפותרים באגים. הדמו האחרון הוא האינטגרציה עם VS Code ו-Claude: JetBrains יודע איזה קובץ פתוח ואיזו שורה מסומנת. בנוסף, הוא בודק פעם בשבוע את ה-changelog בפרויקט Claude Code של Anthropic ב-GitHub, "because even I can't keep up with it" [19:48].



אינטגרציה עם VS Code: מזהה שורה אחת שסומנה ב-math\_utils.py ומתרגם אותה לספרדית לפי בקשה

כמה קובצי CLAUDE.md בפרויקט: בתיקיה אחת אפשר רק קובץ אחד, אבל אפשר לשים קבצים נוספים בתתי-תיקיות. כברירת מחדל נטען רק הקובץ שבתיקית העבודה, ועוד אחד מתיקית הבית אם יש. קבצים בתתי-תיקיות לא נטענים אוטומטית, כי ב-monorepo של Anthropic זה היה מפוצץ את ה-context. במקום זה Claude מונחה לקרוא אותם כשהוא נתקל בהם והם רלוונטיים. אפשר גם לייבא קבצים אחרים מתוך CLAUDE.md עם @.

הערות inline מיותרות שחוזרות שוב ושוב: לדבריו זו הייתה בעיה של המודל ולא של ה-prompt. כבר ב-3.7 היו ב-prompt הוראות מפורשות נגד הערות, "and despite that, the model just loves to leave comments." [22:49]. ב-Claude 4-הבעיה תוקנה ברובה, וההיענות להוראות השתפרה. לכן המעבר למודל חדש הוא הזדמנות לעבור על CLAUDE.md, להוריד מה שכבר לא נחוץ ולהוסיף מה שחסר.

שיתוף context בין agents: אין לזה מנגנון native. הצוות עדיין מעדיף את "the simple thing that works", כלומר agent אחד.

**Cal Rueb — על שיתוף context בין agents (דוגמת Ticket.md) [24:22]** I would think about trying to write that state to a file and then just kind of like count on the model's ability to just like read files and make sense of them. It's probably the best you can do today

## 7. מה עושים עכשיו

- **לכתוב או לרענן CLAUDE.md:** איך מריצים בדיקות, מפת התיקיות, style guide וכלים פנימיים. אחרי כל החלפת מודל כדאי לקצץ ממנו.
- **לאשר קבוע את הפקודות הבטוחות:** להוסיף להגדרות את פקודות הבדיקה וה-lint שחוזרות כל הזמן, כדי להפסיק לאשר אותן ידנית.
- **להעדיף CLI מתועד על פני MCP:** להתקין gh ודומיו ולתעד אותם. MCP כשאינן ברירה אחרת.
- **להרגיל את עצמכם ל-compact/ול-clear:** לפני שה-context מתמלא, ולא אחרי שהאיכות כבר ירדה.
- **לפתוח משימות בבקשת תוכנית:** "תחקור ותציע אפשרויות, אל תכתוב קבצים", ואחר כך לעקוב אחרי ה-TODO.list.
- **guardrails ל-TDD:** vibe coding, שינויים קטנים, lint ו-typecheck בכל צעד, ו-commits תכופים.
- **לנסות שני מופעים במקביל:** עם handoff דרך קובץ markdown משותף, כמו Ticket.md.

| שאלה פתוחה                                 | למה היא חשובה                                                                           |
|--------------------------------------------|-----------------------------------------------------------------------------------------|
| איך מודדים אם שינוי ב-CLAUDE.md שיפר משהו? | Rueb עובד על evals של הכלי, אבל לא הציע למשתמשים דרך למדוד את האפקט של ה-CLAUDE.md שלהם |

| שאלה פתוחה                                        | למה היא חשובה                                                                       |
|---------------------------------------------------|-------------------------------------------------------------------------------------|
| מתי auto-accept מוסכן?                            | ההרצאה ממליצה להאיץ עם הרשאות, אבל לא מתארת אילו פעולות כדאי להשאיר באישור ידני     |
| מתי CLI עדיף על MCP ומתי לא?                      | ההמלצה נשענת על "ניסיון", בלי קריטריון ברור לכלים פנימיים או לכלים שלא מתועדים היטב |
| האם יבוא מנגנון native לשיתוף context בין agents? | נאמר רק "maybe we'll figure out clever ways". בינתיים זה קובץ משותף                 |
| איך נראה שימוש headless מעבר ל-GitHub Actions?    | זה הכיוון שהצוות "הכי מתלהב ממנו", ולא הוצגה בו אף דוגמה                            |

## 8. מילון מונחים

| מונח                        | משמעות                                                                               |
|-----------------------------|--------------------------------------------------------------------------------------|
| agent                       | מודל עם הוראות וכלים שרץ בלולאה עד שהוא מחליט שסיים                                  |
| agentic search              | חיפוש איטרטיבי בקוד עם Grep, Glob ו-Find, כשכל תוצאה מכוונת את החיפוש הבא            |
| indexing / RAG              | אינדוקס הקוד ל-embeddings ושליפה לפי דמיון. Claude Code לא עושה את זה                |
| CLAUDE.md                   | קובץ הוראות שנטען ל-prompt בתחילת כל session, הזיכרון המשותף של הפרויקט              |
| MCP                         | פרוטוקול לחיבור כלים ושירותים חיצוניים ל-Claude agent. Code הוא גם client וגם server |
| auto-accept mode            | מצב שמופעל ב-Shift+Tab, שבו Claude פועל בלי לבקש אישור לכל פעולה                     |
| context window              | כמות הטוקנים שהמודל רואה בבת אחת. בהרצאה: 200,000                                    |
| clear, /compact/<br>session | איפוס ה-session (נשאר רק CLAUDE.md), או סיכום ה-session כ-handoff שממנו ממשיכים      |
| TODO list                   | רשימת המשימות ש-Claude בונה במשימות גדולות. אפשר לעצור ולתקן אותה                    |
| TDD                         | פיתוח מונחה בדיקות: כותבים בדיקה, משנים מעט, מריצים                                  |
| vibe coding                 | לתת ל-agent לרוץ עם מעט התערבות. בהרצאה: עם guardrails                               |
| worktrees / sub-agents      | עותקי עבודה נפרדים של ה-repo, או agents משנה, כדי להריץ כמה משימות במקביל            |
| headless / SDK              | הפעלה תכנותית של Claude Code בלי ממשק, למשל ב-CI/CD                                  |
| extended thinking           | שלב חשיבה גלוי לפני פעולה. מ-Claude 4 הוא קורה גם בין קריאות לכלים                   |
| monorepo                    | repo יחיד שמכיל פרויקטים רבים, כמו ה-repo של Anthropic                               |