איך Salesforce משתמשת ב-Claude Code

שני מובילי הנדסה ב-Salesforce מסבירים איך הגיעו לפי 2 בתפוקה האפקטיבית, עם פחות באגים לכל שורת קוד, ואיך מודדים את זה

Richard ו-M.K. (Salesforce), בשיחה עם מראיין מ-Anthropic · Claude · בנובמבר 2026
משך: כ-16 דקות · YouTube
סיכום אוטומטי — AI Briefing (aibriefing.dev)

על המסמך הזה

זה ראיון קצר בסדרת Office Hours של Claude. מראיין מ-Anthropic משוחח עם שני מובילי הנדסה ב-Salesforce, שבתמלול מופיעים רק בשמם הפרטי, Richard ו-M.K. (השמות המלאים והתפקידים לא נאמרים בהקלטה). השיחה מכסה שנה של הטמעת Claude Code בכל מחזור הפיתוח: מהפרויקט שהפך את Salesforce ל-headless תוך שבועות, דרך מדד effective output שנבנה עם חוקרים מ-Stanford, ועד skills marketplace, incident agents ובחירת מודל לפי משימה.

הטענה המרכזית היא שאימוץ agents בארגון גדול לא קורה לבד. צריך עקומת בשלות מפורשת שמזיזה מהנדסים מ-prompts פשוטים לעבודה אוטונומית, מדד ROI שאי אפשר לנפח בהערות בקוד, ו-guardrails ו-benchmarks שמוכיחים שהאיכות לא נפגעה. לדבריהם, נפח הקוד כמעט הוכפל ושיעור הבאגים לשורת קוד דווקא ירד.

איך לקרוא את המסמך

- **1. תקציר מנהלים:** הטענות המרכזיות עם חותמות זמן, ומה לא נאמר בשיחה.
- **2. Headless תוך שבועות ומדידת ROI:** פרויקט ה-API coverage ומדד effective output.
- **3. עקומת הבשלות והאימוץ:** הדרגות מ-utterance פשוט ועד אוטונומיה, ואיך דחפו אנשים לטפס בהן.
- **4. Skills marketplace, guardrails ו-benchmarks:** שיתוף skills עם evals, איכות מול נפח, והרצה חוזרת של benchmarks על מודלים חדשים.
- **5. אימות, גודל משימה ובחירת מודל:** proof of work מה-agent, למה עדיף לתת משימות גדולות, והבאג ש-Opus מצא תוך 24 שעות.
- **6. מה לעשות עכשיו:** עצות הדוברים, פעולות מומלצות ושאלות פתוחות.
- **7. מילון מונחים:** המונחים באנגלית שמופיעים במסמך.

הערת מקור המסמך הופק אוטומטית מהתמלול ומהפריימים של ההקלטה. זה ראיון בלי שקפים, ולכן כל העובדות מגיעות מהאודיו. ה-ASR שיבש שמות ומונחים: Claude Code מופיע בתמלול כ-cloud code או quad code, והשמות Claudeforce, A-Expert suite ו-Fable לא ודאיים, וזה מצוין במקומות הרלוונטיים. הציטוטים מועתקים מילה במילה ונבדקו מול חותמת הזמן של הבלוק בתמלול. לא נוסף מידע ממקורות חיצוניים.

1. תקציר מנהלים

Salesforce מציגה הטמעה של Claude Code בכל ה-SDLC, עם שלושה עוגנים: מדד תפוקה שאי אפשר לרמות, עקומת בשלות שמזיזה מהנדסים לעבודה agentic, ותשתית eval שבודקת איכות, skills ומודלים חדשים. רוב המספרים נאמרים בעל פה ובלי פירוט מתודולוגי.

- **Headless תוך שבועות במקום שנה:** הפיכת אפליקציות בנות עשרות שנים ל-headless הייתה מוערכת כפרויקט של חצי שנה עד שנה. לדבריהם הגיעו ל-80%-90% API coverage בכל ה-clouds בתוך פחות מכמה חודשים, בלי ישיבות בין צוותים.
- **ROI נמדד ב-effective output ולא בשורות קוד:** זה ציון שנבנה עם חוקרים מ-Stanford ונותן משקל נמוך לשינויים קוסמטיים והערות, ומשקל גבוה לשינוי לוגיקה. המדד הנגזר, דולר ל-effective output, השתפר פי 2.
- **עקומת בשלות agentic:** תוך חודש-חודשיים הזיזו חלק גדול מהמהנדסים לרמות P3 ו-P4, ולדבריהם התפוקה עלתה פי 2 בכל ה-stacks. היום יש אלפי מהנדסים במצב אוטונומי.
- **אימוץ דרך הדגמות חיות ונתונים אמיתיים:** מהנדסים פותרים בעיות בשידור חי, וכל cloud הציג חדשנות שנבנתה עם agents והגיעה ל-Dreamforce.
- **Incident agent לכל צוות:** agent זיהה שהתקלה בשירות פנימי ולא בשירות שקיבל את ה-page, הפנה את הקריאה לצוות הנכון וחסך להעיר תשעה צוותים.
- **Skills marketplace עם evals:** skills משותפים נבדקים ב-online eval שמריץ שוב sessions עם ה-skill ובלעדיו, כדי להוכיח שיש להם ערך על התוצאה.
- **נפח כמעט כפול, פחות באגים לקוד:** נפח הקוד כמעט הוכפל, ושיעור הבאגים לקוד שנוצר ירד בממוצע. את זה הם מייחסים ל-guardrails ולנתוני SDLC מתויגים באופן אחיד.
- **Benchmarks שמורצים מחדש על כל מודל חדש:** כולל קטגוריית fail to fail, משימות long horizon שמודלים נכשלו בהן עד כה. כשמודל חדש מצליח בהן, זו סיבה לאמץ אותו.
- **הטעות הגדולה: התאמת מודל למשימה:** המודל צריך להתאים לאורך ולמורכבות של המשימה. הדוגמה: Opus איתר תוך 24 שעות באג שניסיונות אנושיים לא מצאו.

מה לא נאמר

- **אין הגדרה מלאה של effective output:** לא ברור איך הציון מחושב, מה בסיס ההשוואה ועל איזה פרק זמן נמדד שיפור פי 2.
- **אין מספרים מוחלטים:** לא נאמר כמה מהנדסים יש בסך הכל, מה עלות הטוקנים, מה שיעור הבאגים לפני ואחרי, ומה גודל ה-benchmark.
- **הדרגות לא מוגדרות במלואן:** מוזכרות P3 ו-P4 ותיאור כללי של השלבים, אבל לא מספר הדרגות ולא הקריטריונים למעבר ביניהן.
- **אין תוצאה של תיקון הבאג:** Opus הציע תיקון, אבל לא נאמר אם התיקון אומת ונפרס.
- **אין דיון בכשלים:** מלבד בחירת מודל לא תואמת, לא תוארו תקלות, דליפות, אירועי אבטחה או פרויקטים שנכשלו.

2. Headless תוך שבועות ומדידת ROI



שלושת המשתתפים בשיחה סביב השולחן: המראיין מ-Anthropic (משמאל) ושני מובילי ההנדסה של Salesforce

הפתיחה היא דוגמה שלדברי הדובר לא הייתה אפשרית לפני Claude Code. Salesforce קיימת עשרות שנים ולא לכל דבר יש API. כדי לאפשר את מה שהדובר מכנה Cloud Force (בתיאור הסרטון Claudeforce, השם לא ודאי), צריך היה להפוך את כל האפליקציות ל-headless: לעבור דף אחרי דף, למפות את ה-APIs ולזהות פערים. הערכה הייתה שזה פרויקט של שישה חודשים עד שנה. הדובר מדגיש שמדובר בקוד קיים בן עשרות שנים ולא בקוד חדש, ומוסיף שכנראה לכל אפליקציה או UX חדשים שהוצגו ב-Dreamforce הייתה מעורבות כלשהי של Claude.

> **Salesforce** And now we are at like 80%, 90% of API coverage for all the clouds in a matter of less than a few months with no meetings across the teams. — **דובר מ-**

לגבי ROI, הבעיה הראשונה הייתה מה בכלל מודדים. בשורות קוד אפשר לרמות בקלות, כי "It can't just be lines of code, because then people, you can just keep adding lots of comments.". לכן נבנה עם חוקרים מ-Stanford ציון effective output: ציון נמוך לשינויים פשוטים או להוספת הערות, וציון גבוה לשינוי לוגיקה או ליצירת דבר חדש. מהציון נגזר מדד דולר ל-effective output, שמאפשר להשוות לפני ואחרי הכנסת AI.

השלב הבא הוא להרחיב את השיטה לפונקציות שאינן קוד. לכל פונקציה בוחרים מדד מתאים, כמו סגירת פריטים או נפח cases, משווים לפני ואחרי הכנסת agents, ומצליבים עם צריכת הטוקנים. אם אין שיפור, שואלים אם משתמשים בכלי לא נכון או שהוא פשוט לא מתאים למשימה.

> **effective output** And that's grown like 2x for us, which has been pretty substantial. — **דובר מ-Salesforce, על דולר ל-**

3. עקומת הבשלות של agents ואימוץ

Claude Code משמש ב-Salesforce לכל מחזור ה-SDLC. בהתחלה האימוץ היה חופשי, ואז הוסיפו מבנה. התובנה שהובילה לזה: גם אחרי דחיפה ארגונית חזקה, אנשים עדיין השתמשו בכלי למשימות ברמה נמוכה ולא חצו את הפער לעבודה שבה ה-agent באמת עובד בשבילם. לכן נבנתה, לדבריהם בשיתוף עם Anthropic, עקומת בשלות agentic. אתגר ה-enablement היה להסביר שהמעבר ל-P3 ו-P4 הוא לא משחק סכום אפס אלא דרך עבודה אחרת.

שלב	מה קורה בו (לפי התיאור בשיחה)
Utterances פשוטים	פקודה כמו "go do this" בלי להגדיר איך נראית תוצאה טובה. עובד במשימות פשוטות ונכשל במשימות long horizon.
הגדרת טוב ורע ותכנון	מגדירים מראש איך נראית הצלחה ואיך נראה כישלון, ובונים תוכנית מקיפה לפני הביצוע.

שלב	מה קורה בו (לפי התיאור בשיחה)
Teams ו-sub agents	נותנים ל-harness לפרק את העבודה למשימות מקבילות, לנהל worktrees ולהגביר את המקביליות.
אוטונומיה ואוטומציה	agents מטפלים במשימות SDLC מקצה לקצה: כשל בדיקה ב-CI/CD מתוקן על ידי agent אחד, ו-agent אחר מבצע code review ו-merge.

> **דובר מ-Salesforce** We started with, obviously we started with everyone at the very beginning, but now we have thousands of people in this kind of autonomous mode.

מה הניע את האימוץ? לדבריהם, השראה ונתונים אמיתיים. אנשים שונים הדגימו את דרך העבודה שלהם ופתרו בעיות בשידור חי, וכל cloud הציג חדשנות שנבנתה עם agents בשבועות במקום חודשים והגיעה כמוצר ל-Dreamforce. ארכיטקטית בצוות סיפרה שכדי להבין קוד חדש היא הייתה צריכה כמה ימים ושיחות עם שלושה או ארבעה צוותים, וה-agent ביטל את הישיבות האלה.

הגל השני צמח מלמטה, סביב ניהול אירועים. היום לכל צוות יש incident agent משלו. בדוגמה שתוארה, ה-page הגיע לשירות החיצוני אבל התקלה הייתה בשירות פנימי. ה-agent כתב ב-Slack את ה-root cause, פתח page לצוות הנכון וציין את איש הקשר שם.

> **דובר מ-Salesforce** Imagine nine teams would have been woken up otherwise.

4. Skills marketplace, guardrails ו-benchmarks

ה-marketplace התחיל כרעיון לשתף את ה-skills שצוותים כתבו. בתמלול השם הוא A-Expert suite, וכנראה מדובר בשיבוש ASR. משם הוא התפתח ל-skills marketplace עם evals שמראים אפקטיביות, כולל online eval שמריץ שוב sessions עם ה-skill ובלעדיו. Claude גם מציע שיפורים ל-skills, למשל שימוש יעיל יותר בטוקנים או הגעה מהירה יותר לתוצאה. ה-marketplace הוא חלק מה-engineering harness ותומך בהיררכיה של skills ברמת כל ההנדסה, ברמת הצוות וברמת scrum team. עם אלפי repos, המטרה היא לאפשר לכל מהנדס לתקן באג בכל מקום בקוד: לדעת אילו repos לבחור, באיזה API surface לגעת ובאיזה workflow להשתמש כדי להוציא PR איכותי בקוד שלא הכיר.

ה-guardrail המרכזי הוא איכות. בחודשים האחרונים נפח הקוד כמעט הוכפל, והחשש היה שיהיו גם פי 2 באגים ופי 2 cases. זה לא קרה. מה שמאפשר את המדידה הוא שכולם עובדים לפי אותו תהליך SDLC סטנדרטי, מערכת הטיקטים מתויגת באופן אחיד והכל מקושר ל-PRs, כך שיש dataset מסודר שאפשר להתחיל ממנו.

> **דובר מ-Salesforce** So we are on the average actually less sort of bugs per code that's being generated while our code volume has spiked.

על אותו dataset בנה הצוות של Richard סט benchmark eval. כשמגיע מודל חדש או skill חדש, מריצים שוב את כל המשימות הקצרות וה-long horizon ב-SDLC ובודקים אם התוצאה טובה או גרועה יותר, כמה זה עולה ואילו יכולות חדשות נפתחו. התוספת החדשה היא קטגוריית fail to fail: משימות שמודלים נכשלו בהן באופן עקבי, בעיקר long horizon. כשמודל חדש מתחיל לעבור אותן, זו בדיוק הסיבה לאמץ אותו: "Plus the new thing we've added is the fail to fail."

5. אימות, גודל משימה ובחירת מודל

המראיין מצטט את Richard: "autonomy without validation is confidence theater". התשובה היא לחפש קריטריונים אובייקטיביים שאפשר לתת עליהם ציון, ולהגדיר מראש איך נראה טוב ואיך נראה רע. רע כולל גם negative test cases: האם שברת משהו אחר? האם הוספת שדות לישות קיימת כשביקשתי schema חדשה? במקום לבקש מה-agent לבדוק את עצמו באופן כללי, מבקשים ממנו ראיות לכל feature חיובי ושלילי, ומטריצה של מה שמצא. אם הראיות לא מספיקות, הוא חוזר לעבודה.

> **Richard (Salesforce)** Show me the evidence of the features. Show me the evidence of the negative features.

לגבי גודל המשימה, התשובה מנוגדת לאינטואיציה: "More story points is usually better.". ההשקעה בהגדרת משימה קטנה לרוב לא שווה את זה. עדיף לחתוך קבוצה גדולה של features ולתת ל-harness ולמודל לפרק אותה למשימות מבודדות. משימה של כמה דקות מתאימה לסשן אינטראקטיבי, וכל השאר נצבר לחבילת עבודה גדולה. בני אדם התרגלו לפרק עבודה ליחידות של 1-3 story points, אבל צוות של agents יכול לעשות את הפירוק בעצמו. מה שקובע את איכות התוצאה הוא התוכנית מראש: "The better the plan upfront, the more output you're going to get, the better output you're going to get.".

הטעות הגדולה בהטמעה, לדבריהם, הייתה אי-התאמה בין מודל למשימה: "Use the right model for the right task.". הקריטריונים הם אורך ומורכבות המשימה. השאלה של המראיין בתמלול היא Fable problem מול Sonnet problem, וכנראה זה שיבוש ASR של שם מודל. הדוגמה שניתנה: באג ידוע במוצר קריטי שכמה חקירות אנושיות לא הצליחו למצוא את ה-root cause שלו. Opus דיבג את הקוד, שינה, קימפל וחזר על התהליך.

> **דובר מ-Salesforce** — And after 24 hours, it found it and it came up with a proposed fix.

> **הנקודה החשובה** הדובר מדגיש שהאימות של התיקון הוא החלק החשוב ביותר. מדובר בתוכנה עסקית קריטית ללקוחות, וטעות עלולה להשפיע על נתונים פיננסיים. הם לא רצו שתיקון של באג קטן יהפוך לבאג גדול שקשה באותה מידה למצוא ולתקן.

6. מה לעשות עכשיו

לסיום שני הדוברים נותנים עצה אחת כל אחד. התשובה הראשונה ("Be bold") מכוונת להנהלה: לעבור מ-system of engagement ל-system of agency, לאפשר חדשנות שלא באה רק מלמעלה, וליצור סביבת למידה שבה אנשים משתפים. התשובה השנייה מכוונת למהנדסים: עם agents אפשר לסגור backlog של דברים שתמיד רצית לעשות, וללמוד מה השתנה בקוד של צוות אחר עוד לפני הפגישה איתו. בתמלול לא ברור לגמרי איזו עצה שייכת למי.

> **דובר מ-Salesforce** — It can't just be top-down. It has to be both top-down and bottom-up with the right guardrails.

- **הגדירו מדד תפוקה שאי אפשר לנפח:** מדד שמבחין בין שינוי לוגיקה לשינוי קוסמטי, עם מדידה לפני ואחרי ומדד עלות לתפוקה.
- **נסחו עקומת בשלות פנימית:** דרגות ברורות מ-prompt פשוט ועד agents אוטונומיים ב-CI/CD, ותכנית enablement שמבוססת על הדגמות חיות.
- **שימו evals על skills משותפים:** הריצו שוב sessions עם ה-skill ובלעדיו לפני שמקדמים אותו ב-marketplace.
- **בנו benchmark מנתוני ה-SDLC שלכם:** טיקטים מתויגים ומקושרים ל-PRs, כולל סט fail to fail לבדיקת מודלים חדשים.
- **דרשו proof of work:** קריטריונים חיוביים ושליליים שאפשר לתת עליהם ציון, ומטריצת ראיות מה-agent במקום self-review כללי.
- **התאימו מודל למשימה:** משימות קצרות במודל קל, משימות long horizon קשות במודל גדול, עם אימות קפדני של התוצאה.

שאלה פתוחה	למה זה חשוב
איך בדיוק מחושב effective output?	בלי הגדרה אי אפשר לשחזר את המדד או להשוות לשיפור פי 2 שדווח.
מה הבסיס לטענה על פחות באגים לקוד?	לא נמסרו מספרים מוחלטים, תקופת מדידה או הגדרה של באג.
כמה דרגות יש בעקומה ומה מבחן המעבר?	מוזכרות רק P3 ו-P4. בלי קריטריונים קשה לאמץ את המודל בארגון אחר.
מה עלות הטוקנים מול החיסכון?	מוזכר מדד דולר ל-effective output, אבל לא נמסרו מספרים.
האם התיקון של Opus אומת ונפרס?	זו דוגמת הדגל להתאמת מודל, ולא ידוע מה קרה בה בסוף.

7. מילון מונחים

מונח	משמעות
Headless	חשיפת הפונקציונליות של אפליקציה דרך APIs, בלי תלות בממשק המשתמש.
Effective output	ציון תפוקה שפותח עם חוקרים מ-Stanford ומדרג קוד לפי ההשפעה שלו ולא לפי מספר השורות.
SDLC / PDLC	מחזור חיי פיתוח התוכנה / מחזור חיי פיתוח המוצר.
Agentic maturity curve	עקומת הבשלות: דרגות השימוש ב-agents, מפקודות פשוטות ועד אוטונומיה (P3, P4 בשיחה).
Utterance	פקודה או בקשה בודדת בשפה טבעית ל-agent.
Sub agents / Worktrees	agents משניים שמבצעים חלקים מהמשימה במקביל, כל אחד בעותק עבודה נפרד של ה-repo.
Harness	שכבת התשתית שמריצה ומתזמרת את ה-agents, למשל Claude Code.
Skills marketplace	מאגר פנימי של skills משותפים, עם evals שמוכיחים את הערך שלהם.
Online eval	הרצה חוזרת של sessions אמיתיים כדי להשוות ביצועים עם שינוי ובלעדיו.
Fail to fail	סט משימות שמודלים נכשלו בהן עד כה. מעבר שלהן מסמן יכולת חדשה.
Long horizon task	משימה ארוכה ומורכבת שדורשת הרבה צעדים רצופים.
Story points	יחידת הערכה של גודל משימה בעבודה Agile.
Incident agent	agent שמנתח תקלה, מאתר root cause ומעביר את הקריאה לצוות הנכון.
System of engagement / agency	מעבר ממערכות שבני אדם מפעילים למערכות שבהן agents פועלים בעצמם.
Opus / Sonnet	מודלים של Claude. Opus הגדול מתאים למשימות long horizon קשות.