AI-DLC בפועל: מתודולוגיה, מדידה ולקחים מהשטח

tlv-dvt203 — סיכום כנס, AWS Summit Tel Aviv 2026

דרור, Specialist Architect, AWS | דנה, Technical Account Manager, AWS | נתנאל, AI Engineering Lead, monday.com

10 בספטמבר 2026 | משך: כ-40 דקות | הופק מהקלטת הכנס

מסלול: AWS for Developers in the AI Era

על המסמך הזה

המסמך מסכם את הסשן tlv-dvt203 מתוך AWS Summit Tel Aviv 2026. זהו סשן משולש: שני דוברים מ-AWS מציגים את מתודולוגיית AI-DLC ואת הדרך למדוד אותה, ולאחריהם מקרה בוחן מלא מ-monday.com. המסמך נבנה מתמלול אוטומטי של ההקלטה ומהסליידים שחולצו מהווידאו, כך שאפשר לחזור לתוכן בלי לצפות שוב בארבעים הדקות. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה נטען בסשן ואילו מספרים גובו.
- **פרקים 2–3 — הבעיה והמדידה:** פרדוקס הפרודוקטיביות, ומדד ה-Cost to Serve שבו AWS משתמשת.
- **פרקים 4–7 — המתודולוגיה:** מהו AI-DLC, ארבעת השלבים, שכבת ה-Harness, והדמו החי ב-Kiro.
- **פרקים 8–11 — monday.com:** שלושת השלבים לארגון AI-native, הסוכנים בשטח, הארכיטקטורה והמספרים.
- **פרק 12 ו"מה לוקחים מכאן":** איך מתחילים מחר בבוקר, ומה כדאי לאמץ ובאיזה סדר.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. ההרצאה נישאה בעברית עם מינוח טכני באנגלית, והתמלול משבש חלק מהמונחים הלועזיים (למשל "ארנס" = Harness, "קלוד קוד" = Claude Code, "פיאר" = Pull Request). המונחים נורמלו לאנגלית בגוף המסמך. הציטוטים מובאים כלשונם מהתמלול ואומתו מול חותמת הזמן המצוינת; מספרים שהופיעו על הסליידים סומנו ככאלה. שמות הדוברים מופיעים כפי שנאמרו בהקלטה, ללא שמות משפחה.

1. תקציר מנהלים

הסשן טוען טענה אחת מרכזית: כלי AI לפיתוח, בפני עצמם, לא מזיזים את המחט הארגונית. מה שמזיז אותה הוא מתודולוגיה שמארגנת את כל מחזור החיים סביב סוכנים, ומערכת מדידה שיודעת להוכיח זאת במספר אחד. החלק השני של הסשן הוא ההוכחה: monday.com מציגה מספרים מארגון פיתוח קיים בן יותר מעשור.

- **הפרדוקס אמיתי ומדיד.** המפתחים כותבים הרבה יותר קוד, אבל כמות הפיצ'רים שמגיעה ללקוח לא משתנה. דרור: "אנחנו קונים את הכלים המפתחים כותבים פי שלוש ויותר קוד אולי אפילו יותר אבל כחברה אנחנו לא מרגישים שאנחנו מתקדמים יותר מהר" [1:33].
- **אימוץ AI לבדו נתן ל-AWS שיפור של כ-16%.** לאורך שנה, על מאות צוותי פיתוח: עלייה של 18.3% ב-throughput, ירידה של 30.4% בהתערבויות ב-CI/CD, ירידה של 32.5% בבאגים בפרודקשן, ובמדד המאוחד Cost to Serve for Software — שיפור של 15.9% [6:09], [7:46].
- **AI-DLC היא ספריית קוד פתוח, לא מצגת.** מסגרת עבודה בשלוש שכבות מעל ה-LLM: סוכנים, ואז Harness ארגוני שמכיל best practices, hooks, tools ו-agents מובנים. לפי דנה: "למעלה מאלפי הורדות ברמה היומית" ועבודה עם "מעל 600 לקוחות בשנה האחרונה" [13:55].
- **המפתח הוא Reviewer Agent לכל Lead Agent.** השיפור האחרון במתודולוגיה הוא צימוד כל סוכן לסוכן-מבקר, כך שנושאים טריוויאליים עוברים לשלב הבא אוטומטית ורק החלטות שדורשות שיפוט אנושי מגיעות לצוות [15:26].
- **ב-monday.com הסוכנים כבר בעלי חלק מהותי מהקוד.** יותר מ-40% מה-PRים שנכנסים לפרודקשן נוצרים על ידי סוכנים (הסלייד מראה 45% בספטמבר 2026), יותר מ-50% מה-PRים מאושרים ללא הערה, וה-throughput למפתח יותר מהוכפל [31:11], [34:13].
- **צוואר הבקבוק זז, הוא לא נעלם.** כשה-execution כמעט חינמי, הכאב עובר ל-product alignment מצד אחד ול-validation & release מצד שני: "זה שאנחנו עכשיו יכולים לפתח מאות או אלפי פיצ'רים בשבוע, זה לא אומר שהלקוחות שלנו מסוגלים לקבל את הדבר הזה" [34:13].

2. הפרדוקס: יותר קוד, אותה תפוקה

דרור פתח בשלוש שאלות שכל ארגון פיתוח מתמודד איתן היום: איך מכניסים AI לארגון שמפתח כבר 5, 10 או 20 שנה; איך יודעים בכלל שזה עזר; ואיך הופכים לארגון AI-native. לדבריו, מחקר שהוצג מראה שהדרך היחידה להשתפר בעשרה עד חמישה-עשר אחוזים בפיתוח תוכנה עוברת דרך AI — ואותו מחקר גם חושף את הפרדוקס: הצוותים מרגישים פרודוקטיביים הרבה יותר, אבל בסוף האיטרציה מגיעה לצד השני אותה כמות פיצ'רים.

שתי הגישות שנכשלות באופן שונה

הגישה הראשונה היא AI אוטונומי: זורקים משימה, מקבלים תוצר. דרור טוען שהיא בדרך כלל לא עובדת — ולא באשמת המודלים. "אין לי כאן שום בלנס, אין לי שום דרך לתקן את מה שקורה תוך כדי פעולה", וחמור מכך, מאבדים שתי תכונות יסוד של הנדסת תוכנה.

> **— דרור, [3:04]** אני מאבד אחריות על הקוד ואני מאבד את הבעלות על הקוד או במילים אחרות מי הולך לתקן בג בשתיים בלילה

הגישה השנייה היא AI כעוזר צמוד: המפתח הופך ממבצע למנהל, ונמצא בשליטה בכל שלב. החיסרון שלה הוא תקרה — היא מוגבלת במספר האנשים בארגון. אבל החיסרון האמיתי, לפי דרור, הוא שהיא מייצרת עומס במקום אחר: הצוות משתפר, אבל הדרישות לא נכנסות לקוד באותו קצב, וקוד שנכנס ל-source control לא בהכרח מגיע לפרודקשן כי האופרציה לא מתקדמת באותה מהירות [4:37].

3. איך בכלל מודדים שיפור?

החלק הזה הוא לדעתי החלק השימושי ביותר בחצי הראשון של הסשן, כי הוא נותן נוסחה שאפשר ליישם. דרור פותח בשלילה של המדדים הנאיביים:

> **— דרור, [4:37]** לספור שורות קוד לא באמת אומר לי כלום על פרודקטיביות

ובצורה דומה, לספור שורות קוד שכתב AI יגיד משהו על שיעור האימוץ של הכלי, אבל לא על התוצר שהגיע ללקוח. מתודולוגיות כמו DORA ו-SPACE נועדו למדוד בדיוק את הדברים הנכונים — cycle time ואיכות — ומי שכבר הטמיע אותן, מוטב שימשיך. אבל הטמעה שלהן לוקחת זמן, ולכן מה שעובד בפועל הוא לבחור KPI מייצג. הקושי המובנה, במילותיו: "מאוד קשה למדוד מתי לקוח קיבל פיצ'ר, ומאוד קשה למדוד בגים שלא נכתבו" [4:37] — ולכן מחפשים משתנים שיש להם קורלציה לדבר האמיתי.

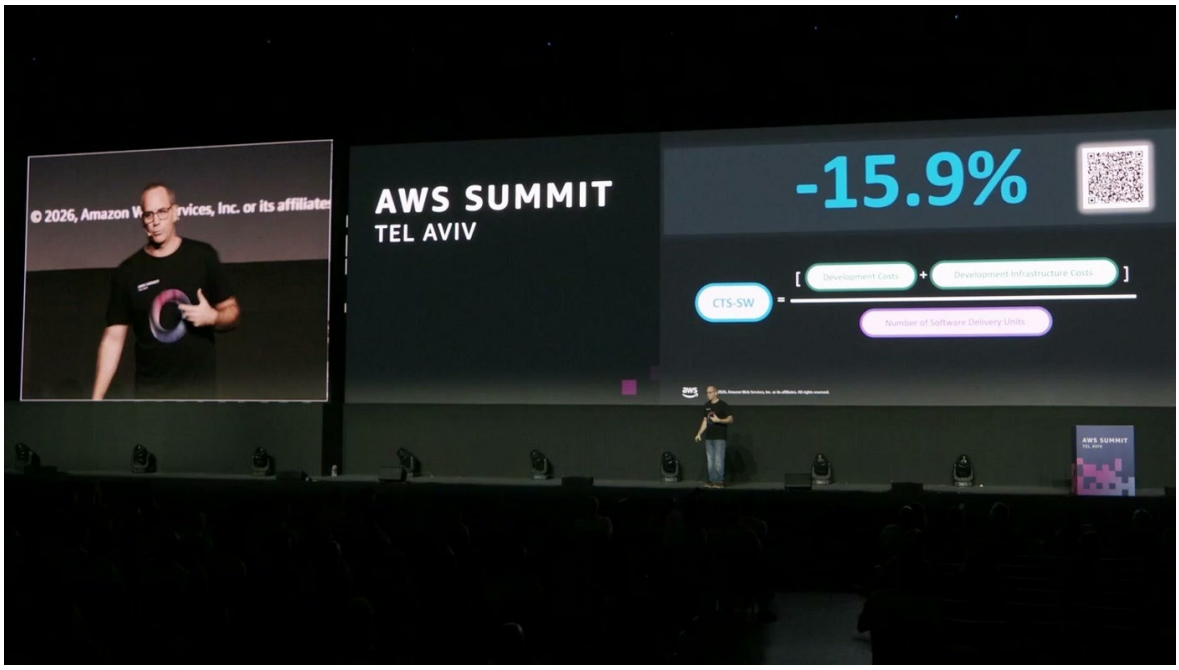
> **לקח מתודולוגי** לפי דרור, ב-AWS גילו שלא אפקטיבי להשוות בין שני צוותי פיתוח, בין אנשים באותו צוות, או בין משימות באותו פרויקט. מה שעובד הוא לקבוע KPI, למדוד baseline, ואז למדוד את אותו צוות מול עצמו לאורך חודשים [6:09].

שלושת ה-KPI שנמדדו ב-AWS

מדד	מה נמדד	השינוי לאחר שנה
Throughput	כמה פעמים בשבוע חבר צוות הביא קוד שהגיע לפרודקשן	+18.3%
התערבות ב-CI/CD	כמה פעמים היה צריך להתערב ידנית בבילד האוטומטי	−30.4%
באגים בפרודקשן	אירועים שהצריכו העלאת מהנדס באמצע הלילה	−32.5%

Cost to Serve for Software — המספר האחד

שלושה מדדים נותנים תחושה, לא הכרעה. כדי לקבל מספר יחיד, AWS שאלה רעיון מ-Amazon.com מעולם המשלוחים — Cost to Serve, כמה מהר הגיעה החבילה ללקוח וכמה זה עלה — והתאימה אותו לפיתוח תוכנה. המונה הוא עלויות הפיתוח (מפתחים וכלים) ועלויות תשתית הפיתוח (dev, test, staging, CI/CD). המכנה הוא מספר יחידות הדליברי.



נוסחת CTS-SW על הבמה: סכום עלויות הפיתוח ועלויות תשתית הפיתוח, חלקי מספר יחידות הדליברי — ולצידה התוצאה שנמדדה, 15.9%−.

הגדרת היחידה היא ההחלטה המעניינת: במונוליט, Pull Request אחד שהגיע לפרודקשן הוא יחידה אחת; מיקרו-שירות אחד שנפרס בענן הוא יחידה אחת גם כן [7:46]. הפרשנות פשוטה:

> **— דרור, [7:46]** אם המספר עולה, או שהעלויות שלי עלו, או שדלברתי פחות או שניהם, זה לא טוב

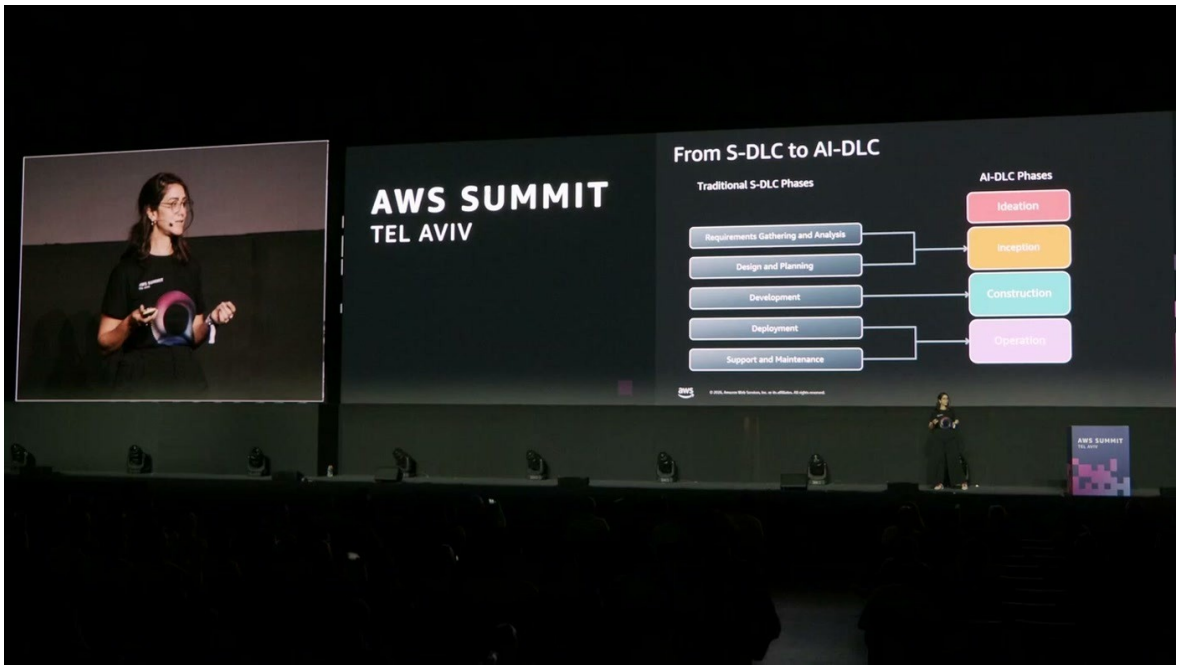
אחרי שנה של אימוץ AI, המדד המאוחד ירד ב-15.9%. וכאן מגיעה הנקודה שמחברת לחלק השני של הסשן: כל זה הוא תוצאה של אימוץ כלים בלבד. כדי להשתפר לא ב-15% אלא בסדר גודל, צריך לשנות את צורת הבנייה עצמה.

4. מהו AI-DLC

דנה הציגה את המתודולוגיה שפותחה בשנה וחצי האחרונות מול לקוחות. AI-DLC הוא משחק מילים על — SDLC Software Development Life Cycle — והוא ספריית קוד פתוח, לא מסמך. מבחינה מבנית זו מסגרת בשלוש שכבות: בליבה יושב ה-LLM; מעליו שכבת הסוכנים, שנותנת למודל יכולת לפנות ל-MCP, לקרוא ל-tools ולפעול; ומעליהם ה-Harness.

> **— דנה, [10:49] (מונחים נורמלו)** Harness זה מה שהארגון שלכם, התשתית הארגונית שלכם להרצת agents, להרצת tools, הרשאות, למי מותר להריץ מה, וכל הידע הארגוני קיים בתוך ה-Harness

AI-DLC מוסיפה ל-Harness הארגוני — אם כבר יש כזה — שכבה של best practices לפיתוח תוכנה, יחד עם tools ו-agents מובנים. השינוי התפיסתי הוא נקודת המגע: בני האדם ניגשים ל-Harness במקום לדבר ישירות עם המודל. כפי שדנה ניסחה זאת: המודל חושב, הסוכן פועל, וה-framework מחליט איך לפעול [10:49].



מיפוי בין חמשת שלבי ה-S-DLC המסורתי לארבעת שלבי AI-DLC: Ideation, Inception, Construction ו-Operation.

ארבעת השלבים

- **Ideation —** עוד לפני שיש פיצ'ר: האם הלקוחות בכלל צריכים את מה שאנחנו עומדים להוסיף, ומה אומר מחקר השוק.
- **Inception —** הגדרת הדרישות, ה-design והתכנון של איך מפתחים את הפיצ'ר או המוצר.
- **Construction —** ה-execution בפועל, ייצור הקוד.
- **Operation —** ה-deployment וכל התהליך שקורה אחריו.

המעגל שחוזר בכל שלב

בתוך כל אחד מארבעת השלבים רץ אותו מעגל: intent נכנס, ה-AI מייצר תוכנית והסבר, הצוות עונה על השאלות שה-AI מציג ומאשר או מבקש שינויים, ה-AI מייצב את התוכנית, ואז מריץ אותה — ובסוף שלב ורפיקציה שבו הצוות מאשר את התוצרים. "כל המעגל הזה קורה בכל אחד מארבעת השלבים שהראיתי קודם, והוא יקרא טיפה שונה ויקח טיפה זמן שונה" [13:55].

המודל האנושי: Mob Programming ו-Squad

לפי דנה, המודל הזה משנה גם את הרכב הצוות. במקום שרשרת בעלי תפקידים — מוצר, ארכיטקטורה, פיתוח, אופרציה, בדיקות — שבה פיצ'ר ממוצע לוקח שבועות עד חודשים, AI-DLC עובד במודל שהיא מכנה mob programming או squad: צוות שאינו מוותר על אף בעל תפקיד, לרוב באותו חדר פיזי, שעובד יחד מול ה-AI.

> **— דנה, [12:19]** משבועות או חודשים לפיצ'ר ממוצע מתקצר לימים בודדים

> **מספרי אימוץ (כפי שנמסרו מהבמה)** "למעלה מאלפי הורדות ברמה היומית, ועבדנו, עשינו סדנאות ועבדנו עם מעל 600 לקוחות בשנה האחרונה בישראל, באירופה ובארצות הברית ואנחנו רואים שיפור של פי חמש ביכולת הדלבור" [13:55]. פי-חמש כאן מוגדר במונחי זמן: מה שלקח חודשים לוקח ימים עד שבועות.

5. השיפור האחרון: Reviewer Agent לכל Agent

החלק הזה הוא לקח מהשטח, לא תכנון מקורי. בגרסה הראשונה, כשמוסרים פיצ'ר ל-AI מופעלים כמה וכמה סוכנים במקביל, ונוצרים מסמכי requirements ומסמכי Markdown עם המון מידע — והמון שאלות שמגיעות לצוות. התוצאה הייתה צוואר בקבוק אנושי: הצוות מקבל יותר מדי נושאים לדיון, וזה לוקח יותר מדי זמן [15:26].

הפתרון: פירוק המשימה לתתי-סוכנים כמו קודם, אבל הוספת reviewer לכל אחד מהם.

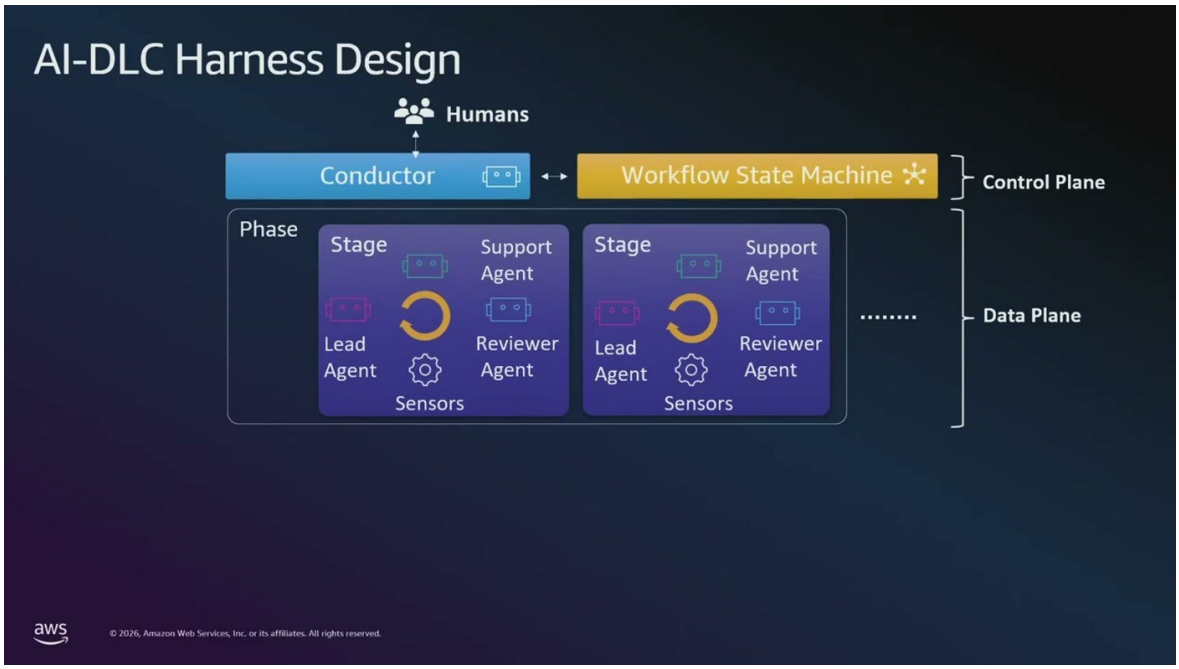
> **— דנה, [15:26]** לכל אג'נט נוצר אג'נט שהוא הריביואר ובעצם בצורה הזאת נושאים שהם טריוויאליים ויש המון כאלה, נושאים שלא מצריכים התערבות אנושית, פשוט עוברים לשלב הבא בצורה אוטומטית

רק מה שדורש הכרעה עיצובית או input קריטי עולה לבני האדם. זה השינוי היחיד שדנה ציינה כמי ש"משמעותית קיצר את התהליך".

6. ארכיטקטורת ה-Harness

בסוף הדמו דנה חזרה למבנה הפנימי. ה-intent מהצוות — high-level או מפורט — מפעיל מכונת מצבים דטרמיניסטית. הנקודה הזו חשובה: הרנדומליות מרוכזת בסוכנים, לא בזרימה.

> **— דנה, [23:09]** כל השלבים האלה שרצים בתהליך הם דטרמיניסטיים בכל שלב ושלב רצים במגביל כמה אגנטים ולכל אגנט יש ריביואר



הפרדת control plane מ-Conductor ו-Workflow State Machine מנהלים, ומתחתם כל Phase מורכב מ-Stages ובכל אחד Lead Agent, Reviewer Agent, Support Agent וסנסורים.

מתחת לכל זה יושבת שכבת coding agents נתמכים — לפי דנה, התמיכה כוללת Claude Code ו-Codex, והרשימה מתועדת ב-GitHub. עבור ארגונים, ההרצה מעל Amazon Bedrock היא הנקודה המעשית:

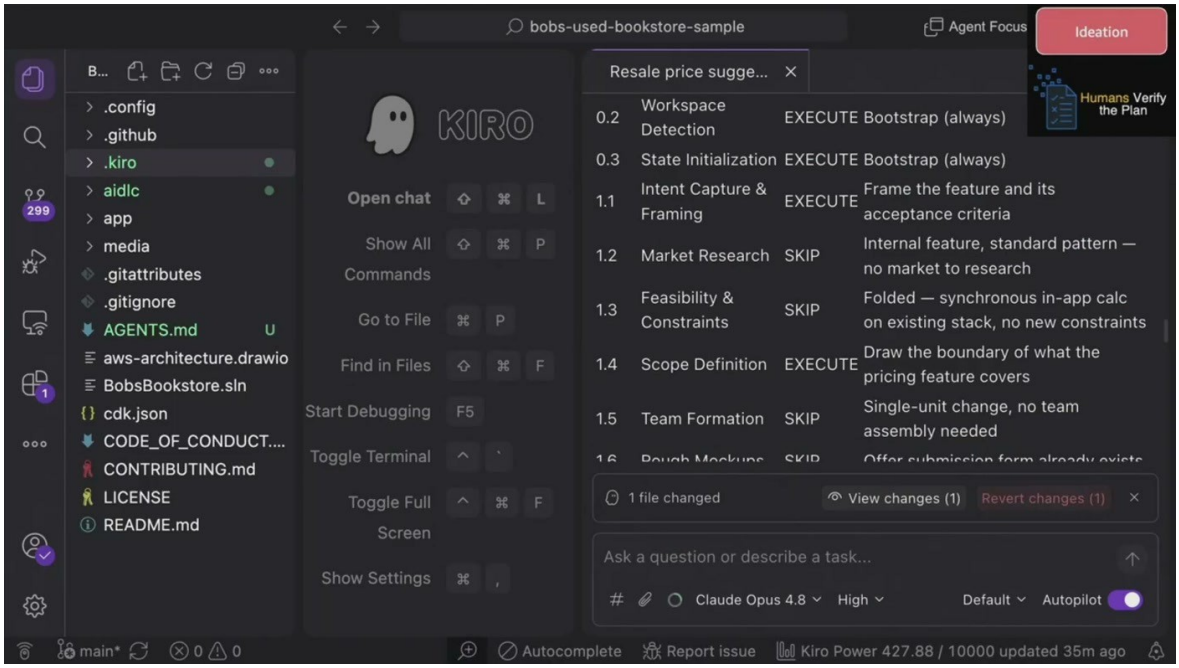
> **— דנה, [23:09]** אנחנו מאפשרים לארגונים לרוץ מעל אמזון בדרוק שמאפשר סקיורטי וקומפליינס ואפשרות לנהל את הצריכת טוקנים

צוואר הבקבוק שזז לוורפיקציה

מ-Goal ל-Production: צוות סוכנים מייצר, וצוות סוכנים שני מוודא — שני הצדדים מקבילים, ונקודת החנק עוברת בין השלבים.

7. הדמו: AI-DLC על קודבייס קיים

דנה הריצה את ה-framework בתוך Kiro, סביבת הפיתוח של AWS, על פרויקט קיים — אפליקציית חנות ספרים — ולא על greenfield. ה-intent הראשון שניתן היה high-level, וה-AI החזיר מיד תוכנית: לא תוכנית לפיצ'ר, אלא תוכנית להרצת ארבעת השלבים עצמם ותתי-השלבים בכל אחד. הספרייה מגיעה עם agents, רשימה ארוכה של hooks ורשימת tools מובנים [17:00].

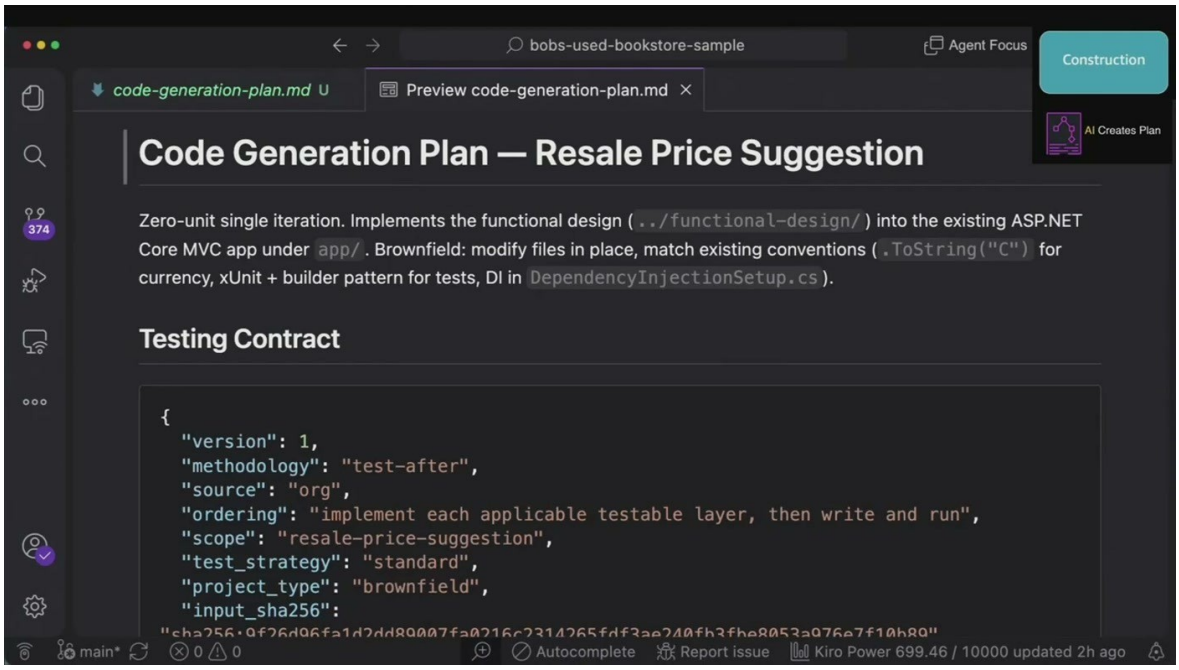


שלב ה-Ideation בתוך Kiro: הסוכן מציג לכל תת-שלב EXECUTE או SKIP עם נימוק — "Internal — Market Research — SKIP — "feature, standard pattern, no market to research" — והתווית מימין מסמנת שהאישור האנושי הוא על התוכנית.

כאן הודגם בדיוק המקום שבו האדם מתערב: בדוגמה, ה-AI תכנן לדלג על שלב יצירת ה-stories, והצוות כתב לו שלא לדלג — "זה שלב קריטי" — וקיבל תוכנית מעודכנת הכוללת reverse engineering ו-story creation [18:32]. אחרי האישור נוצרת היררכיית תיקיות לשלבים, ותחת כל אחת קבצי Markdown שמתארים את הקוד, הסביבה וה-requirements.

Inception: שאלות במקום ניחושים

בשלב ה-Inception ה-AI מציג סט שאלות, לכל שאלה תשובות אפשריות ולעיתים המלצה. דנה הדגישה שזה הרגע שבו ה-squad רלוונטי: איש מוצר, ארכיטקט ומפתח יושבים יחד, על חלק מהשאלות דנים, על חלק מדלגים, ואפשר לענות בחופשיות — למשל "קומבינציה של תשובה אחת, שתיים ושלוש" [20:04]. התוצר הוא מסמך requirements מלא, ואחריו stories עם acceptance criteria שאפשר לייצא ל-Jira או להשאיר במקום [21:38].



תוצר שלב ה-Construction לפני הרצה: Code Generation Plan עם Testing Contract מפורש בפורמט JSON — methodology, ordering, test strategy ו-input hash — שהצוות מאשר לפני שנכתבת שורת קוד.

רק אחרי אישור המסמך הזה ה-AI מריץ בפועל, מייצר את הקומפוננטות ואת הטסטים, ומסכם את מה שנוצר כדי שאפשר יהיה להעביר את ההקשר לחבר צוות אחר ולהמשיך משם [23:09].

8. monday.com: משלושה שלבים לארגון AI-native

נתנאל, המוביל את תחום ה-AI engineering ב-monday.com, הציג את המעבר של החברה מ-AI-assisted ל-AI-native. הנתונים שהוא פתח בהם חשובים כדי לשפוט את שאר המספרים: זה לא ניסוי מבודד.

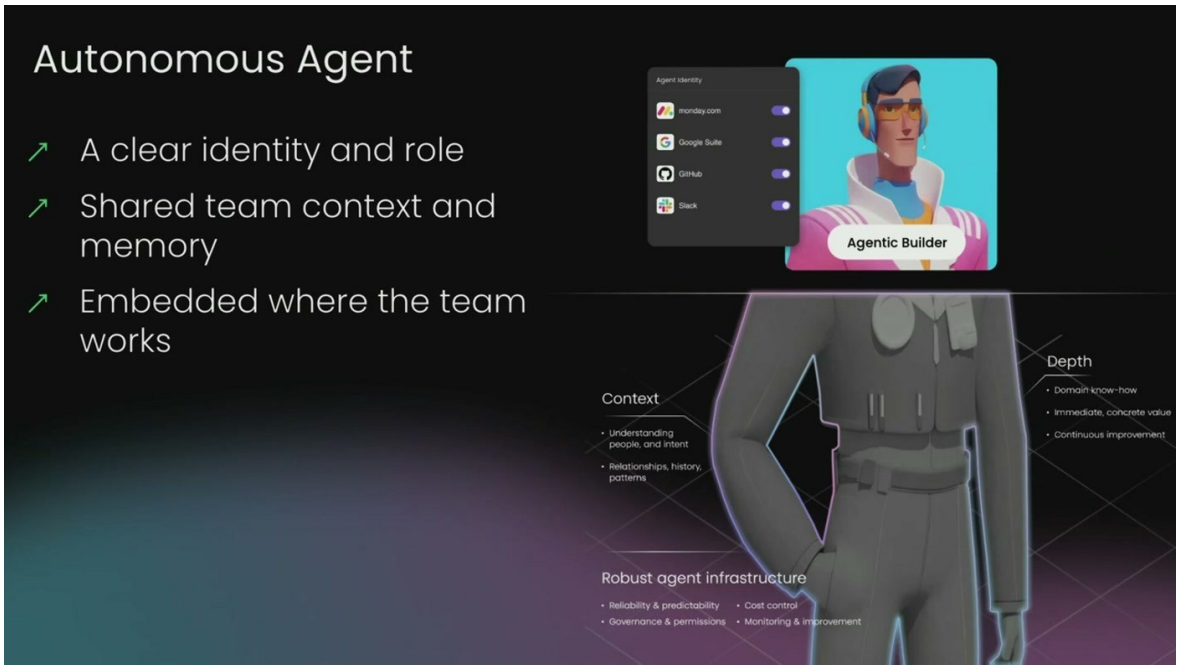
- **הסקייל:** "אנחנו 650 בילדרס, 700 מייקרוסרוויסס בפרודקשן, הכל כמובן על AWS" [24:46].
- **הוותק:** "קודבייס של יותר מ-10 שנים, יותר מ-2 מיליון יוזרים, 150 אלף אקאונטס" [24:46].
- **הקושי:** "לגמרי לא גרינפילד, מערכת פרודקשן סופר מורכבת שרצה ומשרתת במולטי ריג'ן" [24:46].
- **מבנה הצוות:** מודל של builders — צוותים המורכבים מפרודקט, דיזיין, מפתחים ואנליסטים [24:46].

לפי נתנאל, speed of execution תמיד היה ערך תרבותי בחברה, אבל ה-execution עצמו תמיד היה צוואר הבקבוק המשמעותי. בתחילת 2026, ב-company kickoff, הוחלט ללכת all-in על AI — ולא רק בכלי הפיתוח: שינוי המוצר מ-work management, מוצר שהעבודה מתנהלת בו, ל-AI work platform, מוצר שהעבודה נעשית בו בפועל. זה גורר סוכנים במרכז המוצר, ניווט חדש, onboarding חדש ומודל תמחור חדש [26:20]. ההבנה שנגזרה מכך: כדי לבנות מוצר כזה, צריך להיות AI-native בעצמך.

שלושת השלבים, עם מספר לכל שלב

שלב	צורת העבודה	התוצאה שדווחה
Level 1 — Pair programming	מפתח פותח Cursor או Kiro, נותן prompt באנגלית פשוטה, ומקבל קוד. כך עבדו עד תחילת 2026.	+14% ב-PR throughput
Level 2 — Agentic mode	מעבר מ-single-threaded ל-multi-session, עם skills ו-sub-agents. ה-"No Code Month" — חודש בלי IDE, רק CLI עם Claude Code.	+19% נוספים
Level 3 — Autonomous agents	סוכנים שרצים remote, מקבלים משימה ולוקחים אותה end-to-end עד פרודקשן.	ראו פרק 10

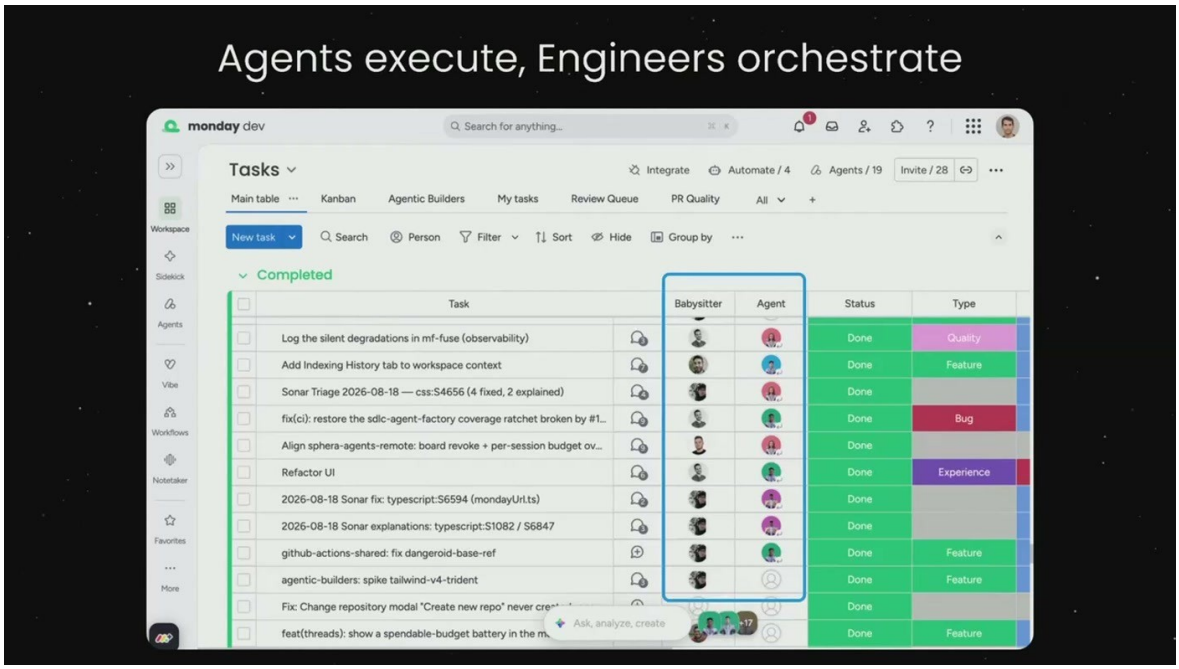
כדאי לשים לב לכנות בסיפור של Level 2: החודש נפתח למחרת אירוע ביטחוני, "אף אחד לא באמת כתב קוד, ציפינו שהאיג'נטים יעשו את זה, לא ממש קרה באותה תקופה" [27:55]. המעבר הושלם במחזור שאחריו.



שלושת המאפיינים שמגדירים סוכן אוטונומי במודל של monday: זהות ותפקיד ברורים, הקשר וזיכרון משותפים עם הצוות, ונוכחות במקום שבו הצוות כבר עובד.

המשמעות המעשית: לסוכן אוטונומי ב-monday יש identity משלו — user אמיתי בכל מערכת שהחברה עובדת בה, כולל monday עצמה, Google, GitHub ו-Slack. הוא חולק את אותו team context, אותו סט goals ואותם plugins כמו המפתחים, ונמצא איפה שהצוות נמצא [29:35].

9. הסוכנים בשטח: ארבע דוגמאות

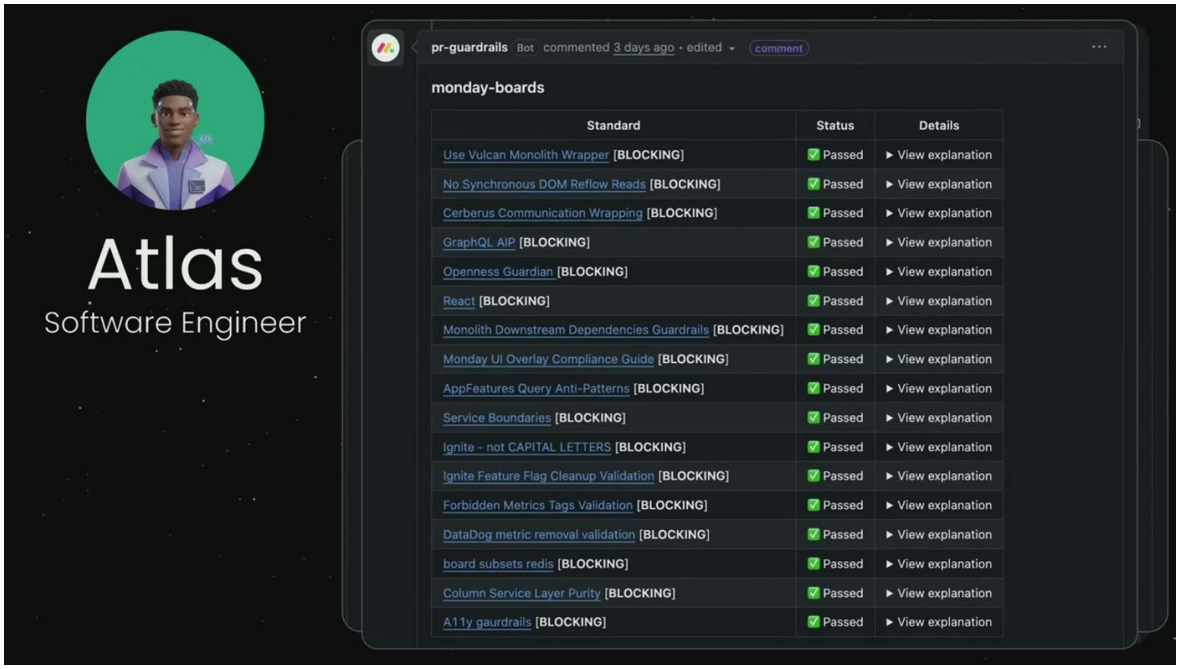


לוח משימות ב-monday עם שתי עמודות חדשות — Agent, ו-Babysitter שמחזיק את האחראי האנושי; שיוך משימה לסוכן שולח webhook והוא מתחיל לעבוד.

— נתנאל, [29:35] יש עמודה של בייביסיטר, שזה המקצוע החדש שלנו, זה בייביסיט על העבודה של האג'נטים

Atlas — מ-task ל-PR

Atlas הוא ה-remote agent הראשון שפותח. הוא לוקח משימה והופך אותה ל-Pull Request, מצרף ל-PR צילום מסך של לפני ואחרי, וכשצריך גם וידאו. אחריו רץ סוכן שני, שנקרא בינתיים בשם גנרי PR Guardrails, ומריץ בדיקה מול הסטנדרטים שהוגדרו ב-monday לאורך השנים — ברמת הריפוזיטורי הספציפי, ברמת הצוות או הקבוצה, וברמת החברה [31:11].



פלט ה-PR Guardrails כתגובת בוט על ה-PR: טבלה של סטנדרטים ארגוניים מסומנים BLOCKING, כל אחד עם סטטוס והסבר שאפשר לפתוח.

— נתנאל, [31:11] כבר היום אנחנו רואים שיותר מ-50% מה-PRים שלנו מאושרים ללא שיהיו מנוסיף עליהם הערה. אנחנו שואפים להגיע למצב של אוטו מרדש

היעד המוצהר הוא auto-merge: שלא יידרשו אנשים שיעברו על הקוד בדרך לפרודקשן. לפי נתנאל, יש כבר סוכנים וצוותים שעובדים כך, והאתגר הנוכחי הוא לעשות זאת בסקייל משמעותי יותר.

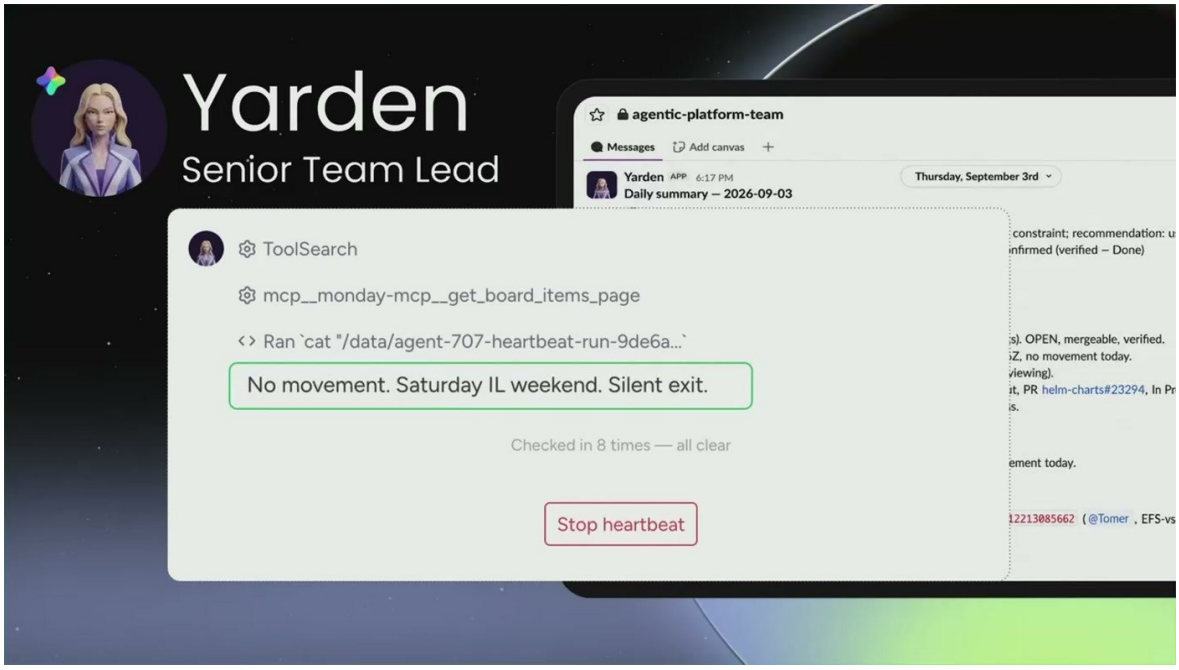
Spike — חקירת פרודקשן

Spike הוא סוכן שמתויג ב-Slack כשמישהו מרגיש בעיה. בדוגמה שהוצגה, אדם מדווח על איטיות ב-page load, אחד הדירקטורים מתייג את Spike, ותוך דקות מתקבל דוח מפורט עם screenshot ופינפוינט לשלב שבו הבעיה מבודדת ולנקודת הזמן שבה התחילה. נתנאל: "אני מניח שיש פה הרבה אנשים שניהלו פרודקשן ויודעים מה זה אומר להגיע לנקודה כזאת" [31:11].

Yarden — Senior Team Lead

כשמספר הסוכנים גדל, ה-babysitting עצמו הפך לעומס. התשובה הייתה סוכן-מנהל: Yarden שולחת כל בוקר סיכום של מה נעשה, מה ב-risk ומה צריך לדעת. הדוגמה שסיפרה את הסיפור הטוב ביותר הייתה תיקון התנהגות:

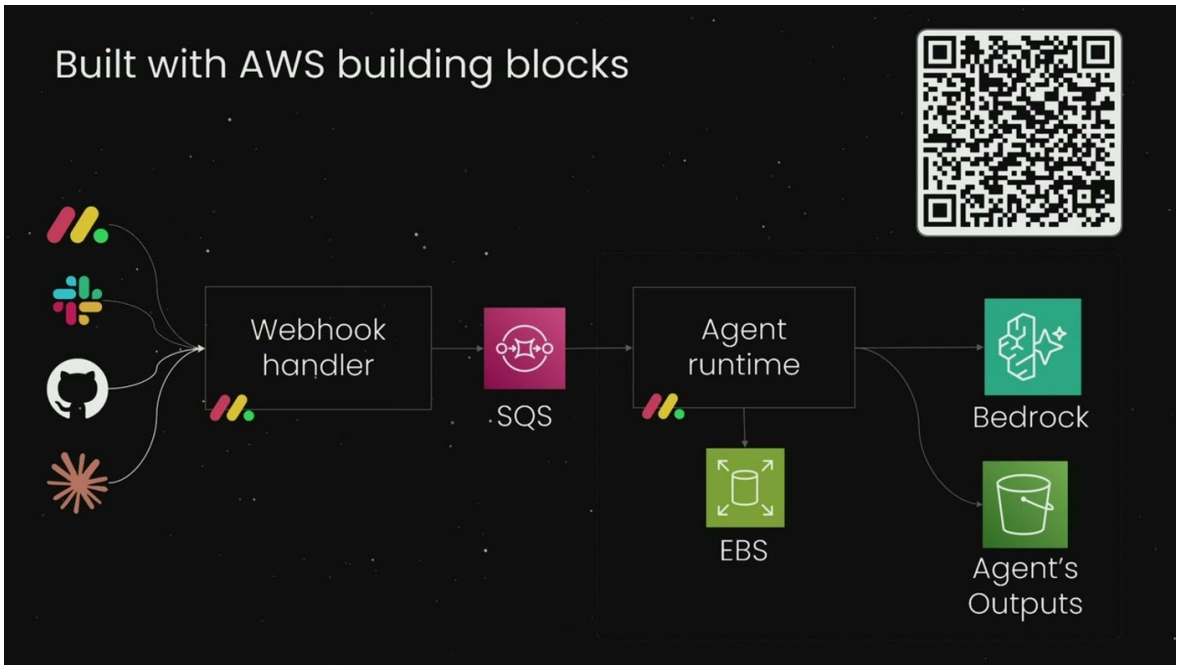
— נתנאל, [32:41] היא שלחה לנו לא מעט הודעות בחגים בסופי שבוע, אמרנו לה תתעני את חגי ישראל, תבין אם מה הוויקנד שלנו, אז היא טענה וכמו שניתן לראות פה, היא רואה שוויקנד אז היא מדלגת



ה-heartbeat של Yarden מדלג מרצונו: קריאות ToolSearch ו-MCP, ואז ההחלטה "No movement. Saturday IL weekend. Silent exit". אחרי שמונה בדיקות.

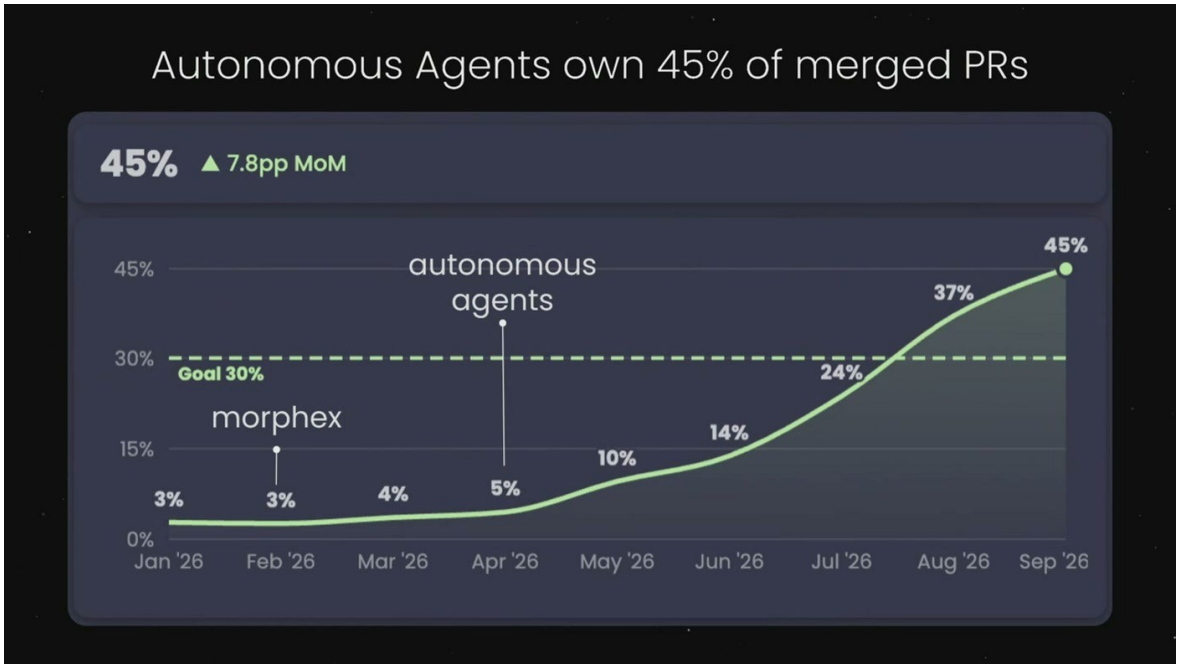
נתנאל הוסיף שעם הזמן הסוכנת פיתחה אישיות — כשתויגה על בעיה מחוץ לשעות, השיבה שהיא לא זמינה והעבירה את המשימה לסוכן אחר [32:41]. מסגור המספרים חשוב כאן: "יש היום יותר מאלפיים אייג'נטים כאלה שרצים בארגון של הבילדרס", אך מיד לאחר מכן — "לא כולם הם באמת משמעותיים... זה סופר מאתגר להגיע למצב שאייג'נט באמת יכול לקחת משימה end to end" [32:41].

10. הארכיטקטורה והמספרים



מערכת הסוכנים של monday ברכיבי AWS: אירועים מ-Slack ומ-GitHub ל-webhook handler, דרך SQS ל-agent runtime, שמריץ מול Bedrock וכותב פלטים ל-EBS ולאחסון ייעודי.

נתנאל הדגיש שהארכיטקטורה "די פשוטה", שהכל בנוי על AWS building blocks, ושגם כאן הכל רץ על Bedrock. הוא הפנה למאמר deep-dive שנכתב יחד עם צוות AgentCore ו-Bedrock [32:41].



עקומת חלקם של הסוכנים האוטונומיים בסך ה-PRים הממוזגים — מ-3% בינואר 2026 ל-45% בספטמבר, עם קו יעד מקווקו ב-30% שנחצה ביולי.

בדברי נתנאל, יותר מ-40% מה-merged PRs נכנסים לפרודקשן על ידי סוכנים, והמספר עולה משבוע לשבוע כי הסוכנים מספקים סקייל "כמעט אינסופי" [34:13]. הסלייד מציג 45% לספטמבר 2026, עלייה של 7.8 נקודות אחוז מהחודש הקודם. ה-PR throughput הכולל, לפי הגרף שהוצג, הגיע ליותר מפי-שניים — כשהתוספת הירוקה היא כולה תרומת הסוכנים.

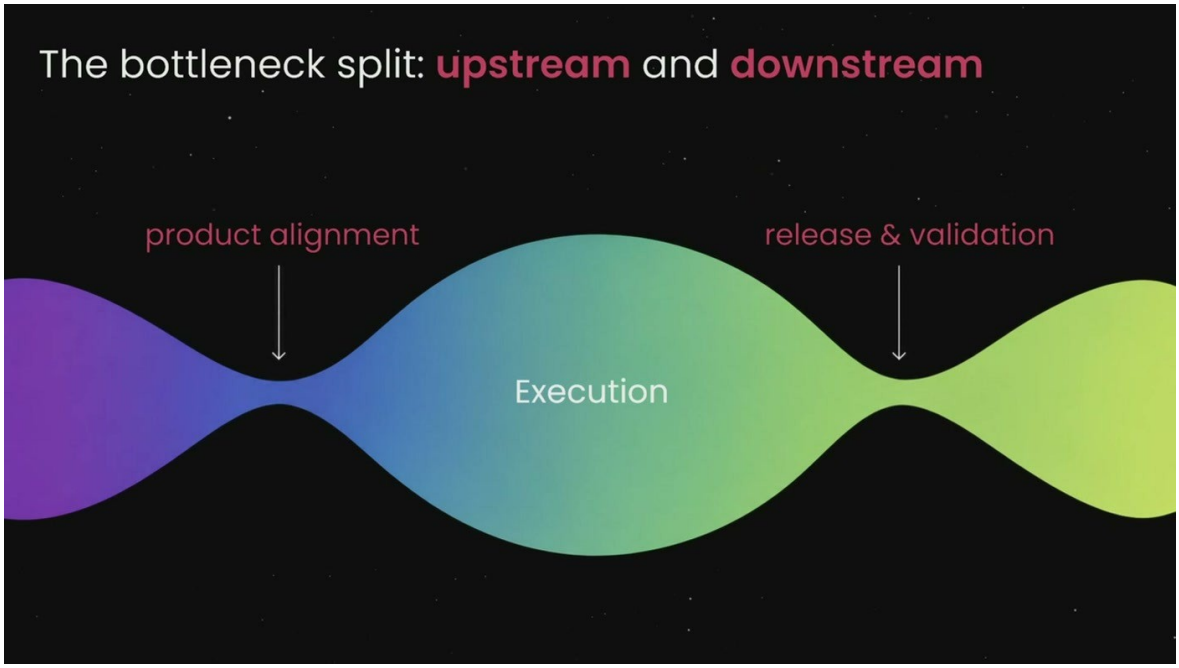
ומה עם האיכות?

זו השאלה שנתנאל צפה מהקהל, והתשובה שלו הייתה מדידה ולא הצהרה: מספר הטיקטים שמגיעים מלקוחות ביחס ל-1,100 PRים.

> **— נתנאל, [34:13]** אנחנו מודדים מספר טיקטים שמגיעים מלקוחות ביחס ל-1100 פיארים, אז אפשר לראות שיש פה שיפור של יותר מ-50 אחוז

לצד זאת דווח שיפור גם ב-average time to resolve. זהו המדד שסוגר את הלולאה עם החלק של דרור: נורמליזציה של תקלות לפי יחידות דליברי, ולא ספירה אבסולוטית.

11. צווארי הבקבוק החדשים



ה-execution מתרחב באמצע ומתכווץ בשני קצותיו: product alignment ב-upstream ו-release & validation ב-downstream הם נקודות החנק שנותרו.

כשה-execution כמעט חדל להיות מגבלה, נחשפו שני צווארי בקבוק חדשים. הראשון הוא product alignment — מה לפתח ומה באמת חשוב ללקוחות. השני הוא release & validation, והוא המעניין יותר כי הוא חיצוני:

> **— נתנאל, [34:13]** זה שאנחנו עכשיו יכולים לפתח מאות או אלפי פיצ'רים בשבוע, זה לא אומר שהלקוחות שלנו מסוגלים לקבל את הדבר הזה

לפי נתנאל, כבר יש לקוחות שמבקשים להאט את הקצב, ובתגובה נפתח פרויקט בשם Enterprise Release — "איך לנהל ריליס בעולם שרץ כל כך מהר" [35:46]. השלב הבא שהוגדר הוא הרחבת צורת העבודה מעבר לארגון הפיתוח: לפרודקט, ואף לתפקידים מחוץ לארגון ה-builders כמו go-to-market וארגון ה-FDE, כך שכולם יעבדו בצורה אחת.

12. איך מתחילים

שלושת הצעדים שהוצגו בסיום — Install את ה-Harness, Learn דרך סדנה, ו-Develop בפיילוט עם קריטריוני הצלחה וביקורת בשערים.

- **Install —** "תלכו לגיטב, תורידו את הספרייה, מכילה סוכנים, skills, hooks לפי כלי הפיתוח החביב עליכם" [37:18].
- **Learn —** סדנה חופשית זמינה באינטרנט, עם הסברים על כל שלב ומה אמורים לראות בו, כדי לפתח מוצר בסביבה בטוחה יחסית [37:18].
- **Develop —** פיילוט בסיכון נמוך: פיצ'ר חדש או פרויקט הבא. לחלופין — לקחת את כל הצוות לחדר אחד ליום או יומיים ולראות מה התהליך מייצר [37:18]. הסלייד מוסיף: הגדירו קריטריוני הצלחה, ובקרו בשערים.

דרור סגר בסיכום שלוש הטענות: איך מודדים פיתוח ואיך יודעים אם באמת נעשינו פרודוקטיביים עכשיו כשמשלמים כסף על כלי AI; המתודולוגיה עצמה; והסיפור של monday מהתחלת האימוץ ועד ארגון AI-native מבוסס סוכנים [35:46].

מה לוקחים מכאן

- **קבעו baseline לפני שקונים עוד כלים.** שלושת ה-KPI שהוצגו — throughput לפרודקשן, התערבויות ב-CI/CD, ואירועי פרודקשן — ניתנים לחישוב מהנתונים שכבר קיימים ב-git וב-CI, ואינם דורשים הטמעת DORA מלאה. בלעדיהם אין דרך לדעת אם ההשקעה עבדה.
- **Cost to Serve for Software הוא המדד שמדבר עם הנהלה.** הוא מאחד עלות ותפוקה למספר אחד, וההגדרה של "יחידת דליברי" (PR שהגיע לפרודקשן, או מיקרו-שירות שנפרס) מספיק פשוטה כדי שאפשר להסכים עליה מראש.
- **השוו צוות מול עצמו לאורך זמן, לא צוות מול צוות.** זו הייתה המלצה מפורשת מהבמה [6:09], והיא גם מנטרלת את ההתנגדות הפוליטית הצפויה למדידה.
- **המנגנון המעניין ביותר הוא Reviewer Agent, לא הסוכן המייצר.** הוא זה שמכריע מה מגיע לאדם ומה לא, והוא מה שהפך את AI-DLC משימושי-בקושי לשמיש. אותו רעיון בדיוק חוזר ב-PR Guardrails של monday: קודיפיקציה של הסטנדרטים הארגוניים כבדיקות חוסמות.
- **דטרמיניזם בזרימה, סטוכסטיות בסוכנים.** מכונת המצבים היא מה שמאפשר גם ממשל וגם חזרתיות. זו התשובה ל"מי אחראי על הקוד" שדרור פתח בה.
- **התכוננו לצוואר הבקבוק הבא לפני שהוא מגיע.** ב-monday ה-execution הפסיק להיות המגבלה, ומה שנחשף היה תיעדוף מוצרי מצד אחד ויכולת הלקוח לספוג שינויים מצד שני. בארגון מוסדר, הצד השני הזה — ולידציה, ריליס וניהול שינוי — יהיה המחסום המוקדם יותר.

- **צעד ראשון מעשי:** התקנת הספרייה מ-GitHub, הרצת הסדנה, ובחירת פיצ'ר יחיד בסיכון נמוך עם קריטריוני הצלחה שהוגדרו מראש — במקום פיילוט רוחבי.

מילון מונחים

מונח	מה זה	כפי שהופיע בתמלול
AI-DLC	AI-Driven Development Life Cycle — מתודולוגיה וספריית קוד פתוח של AWS המארגנת את מחזור חיי הפיתוח סביב סוכנים, בארבעה שלבים.	"AI DLC"
Harness	שכבת התשתית הארגונית להרצת סוכנים וכלים: הרשאות, מי מורשה להריץ מה, והידע הארגוני. AI-DLC מוסיפה מעליה best practices, hooks ו-tools.	"ארנס"
Conductor	רכיב ה-control plane שמתזמר את מכונת המצבים ומתקשר עם בני האדם.	—
Lead / Reviewer Agent	בכל Stage רץ סוכן מוביל וסוכן מבקר; המבקר מחליט אילו החלטות טריוויאליות עוברות הלאה אוטומטית ואילו עולות לאדם.	"ריביואר"
Squad / Mob programming	צוות חוצה-תפקידים (מוצר, דיזיין, ארכיטקטורה, פיתוח) שעובד יחד מול ה-AI, לרוב באותו חדר.	"סקווד"
CTS-SW	Cost to Serve for Software — (עלויות פיתוח + עלויות תשתית פיתוח) חלקי מספר יחידות דליברי.	"קוסט טו סרב"
יחידת דליברי	PR אחד שהגיע לפרודקשן במונוליט, או מיקרו-שירות אחד שנפרס בענן.	—
Kiro	סביבת הפיתוח (IDE) של AWS שבה הודגם ה-framework.	"קירו"
Babysitter	התפקיד האנושי שמלווה סוכן אוטונומי; ב-monday זו עמודה בלוח המשימות.	"בייביסיטר"
PR Guardrails	סוכן שמריץ את הסטנדרטים הארגוניים כבדיקות חוסמות על כל Pull Request.	"פי-אר גארדרי"