מ-IDE לאוטונומיה: צוות ההנדסה החדש

TLV-ARC305 — סיכום כנס, AWS Summit Tel Aviv 2026

Guy Menahem, Solutions Architect, AWS | Alon Gendler, Sr. Solutions Architect Manager, AWS | Gilad Wisney, VP R&D, Reco

10 בספטמבר 2026 | משך: כ-41 דקות | הופק מהקלטת הכנס

סיכום מאת קובי אלמוג

על המסמך הזה

המסמך מסכם את הסשן TLV-ARC305 מתוך AWS Summit Tel Aviv 2026, מסלול Architecting on AWS, ברמה 300. הסשן עוסק בשאלה אחת: איך ארגון פיתוח עובר מ-AI שיושב בתוך ה-IDE ועוזר למפתח, דרך סוכנים שמורצים בהנחיית מפתח, ועד סוכנים שמריצים משימות מקצה לקצה בלי אדם בלולאה — ומה נדרש בתשתית, באבטחה ובממשל בכל אחד מהשלבים.

המסמך נבנה מתמלול אוטומטי של ההקלטה ומהסליידים שחולצו מהווידאו עצמו. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה נטען בסשן ומה ממנו בר-יישום.
- **פרק 2 — המפה:** שלושת שלבי המסע שהמרצים הגדירו, והוא הציר של כל שאר המסמך.
- **פרקים 3–5 — שלושת השלבים:** AI-Assisted, Human-Guided, ו-Autonomous. בכל שלב: מה עובד, מה נשבר, ואיזה שירות נכנס לתמונה.
- **פרק 6 — המקרה של Reco:** VP R&D של חברת אבטחה ישראלית מראה pipeline של שישה סוכנים בפרודקשן.
- פרקים 7–8 — מה לוקחים מכאן, ומילון מונחים לועזיים שהוצגו.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. הסשן נאמר בעברית עם צפיפות גבוהה של מונחים לועזיים, והתמלול משבש חלק מהם — שמות המוצרים והמונחים תוקנו ידנית מול הסליידים שעל המסך. הציטוטים מובאים כלשונם בתמלול, כולל שיבושי הגייה, ואומתו מול חותמת הזמן המצוינת. מספרים ושמות שמופיעים במסמך נלקחו מהסליידים או מדברי המרצים, ולא הושלמו מידע חיצוני.

1. תקציר מנהלים

שלושה דוברים חילקו ביניהם את הסשן: Guy Menahem ו-Alon Gendler מ-AWS, שעובדים מול סטארט-אפים, ו-Gilad Wisney, VP R&D בחברת Reco, שהביא את הצד המעשי. המסר המרכזי אינו שכדאי להשתמש ב-AI לכתיבת קוד — זה כבר נחשב נתון — אלא שהמכפילים האמיתיים מגיעים רק כשמורידים את האדם מהנתיב הקריטי, וזה מחייב תשתית של זהות, קטלוג כלים וגבולות גזרה.

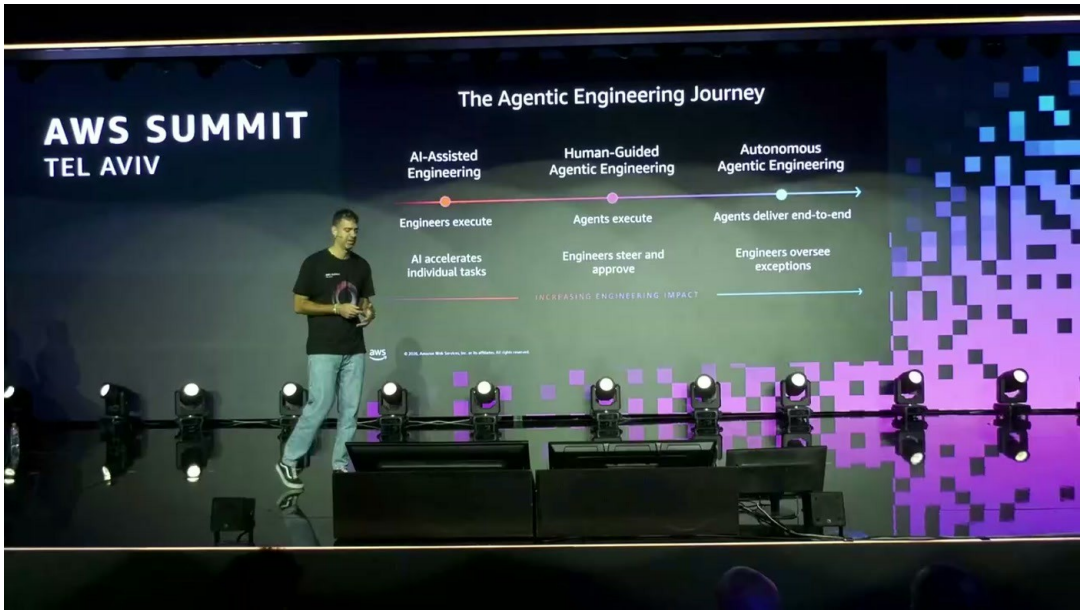
- **המסע מחולק לשלושה שלבים מוגדרים.** AI-Assisted Engineering (המהנדס מבצע, ה-AI מאיץ משימות), Human-Guided Agentic Engineering (הסוכן מבצע, המהנדס מכוון ומאשר), ו-Autonomous Agentic Engineering (הסוכן מספק מקצה לקצה, המהנדס מטפל בחריגים). זה הסליידים והציר של כל הסשן [1:30, 22:59].
- **האדם בלולאה הוא צוואר הבקבוק, לא המודל.** "אנחנו מסתמכים על אנשים בתהליך, הם לא עובדים בלילה צריכים ללכת לישון מדי פעם" [21:26] — ולצד זה Quality Inconsistency, כי כל מפתח בוחר מודל אחר ומגדיר איכות אחרת.
- **התפשטות ספונטנית של כלים היא בעיית אבטחה.** מפתחים מורידים skills ו-MCPs לפי טעם אישי; שלושת האתגרים שהוצגו הם Security, Consistency ו-Visibility, והמטרה היא להפוך את הכלים "מקלי פרודקטיביטי אינדיבידואליים לממש תשתית ארגונית בטוחה" [9:07].
- **מספר ה-MCPs הפומביים הופך allow-list לגישה נאיבית.** "יש היום יותר מ-50 אלף פאבליק MCPs לטוב ולרע" [15:18]. התשובה שהוצגה: AWS Agent Registry כקטלוג ארגוני עם חיפוש סמנטי, ו-Amazon Bedrock AgentCore Gateway כנקודת כניסה יחידה שמזהה את הסוכן ומנהלת את הטוקנים שלו.
- **אוטונומיה נבנית דרך harness, לא דרך פרומפט טוב יותר.** "harness זה כמו תבנית" [22:59] — system prompt קבוע, מודל קבוע, MCPs וskills מחוברים מראש, זיכרון, והרצה ב-CLI במצב non-interactive. אותו harness מורץ ב-Amazon EKS או ב-Bedrock AgentCore Runtime.
- **Reco מריצה את זה בפרודקשן, עם גבולות ברורים.** pipeline של שישה סוכנים מ-Ticket Refinement ועד E2E Regression; סוכנים בסביבת הלקוח מבודדים לחלוטין, האדם נשאר רק בלחיצת ה-merge, והסוכנים מטפלים כרגע בטיקטים בדרגת מורכבות 1–2 מתוך 5 [36:45].

מה שלא נאמר

הסשן לא הציג מדידה של שיעור הצלחה, עלות טוקנים בפועל או אחוז PR-ים שנדחו. המספר היחיד שנטען לגבי תפוקה נאמר בפתיחה כטווח אפשרי — "יכול להגיע עד פי חמש" [0:00] — בלי מקור או הקשר מדידה. מי שמחפש ROI מוכח לא ימצא אותו כאן; מה שכן יש הוא ארכיטקטורה מפורטת ומקרה שימוש אמיתי אחד.

2. המפה: שלושה שלבים בדרך לאוטונומיה

הסליד שחוזר לאורך כל הסשן מגדיר שלוש תחנות, כשהציר התחתון הוא Increasing Engineering Impact. ההבחנה בין התחנות אינה בכמות ה-AI אלא בשאלה מי מבצע ומי מפקח.



שלוש התחנות: המהנדס מבצע, הסוכן מבצע תחת הכוונה, והסוכן מספק מקצה לקצה בזמן שהמהנדס מפקח על חריגים

שלב	מי מבצע	תפקיד המהנדס
AI-Assisted Engineering	Engineers execute	ה-AI מאיץ משימות בודדות
Human-Guided Agentic Engineering	Agents execute	Engineers steer and approve
Autonomous Agentic Engineering	Agents deliver end-to-end	Engineers oversee exceptions

הפתיחה מסגרה את המוטיבציה: "אנחנו גם רואים שהמספרים או היכולות שלנו להגיע לדברים משמעותיים, יכול להגיע עד פי חמש" [0:00], אבל מיד אחר כך הסתייגות שהיא למעשה כותרת הסשן — "אנחנו צריכים להיות מאוד מאוד חכמים ולאפשר לאג'נטים שלנו לעשות יותר ויותר עבודה ולקחת יותר ויותר אחריות" [1:30].

3. שלב ראשון — AI-Assisted Engineering

השלב שרוב הארגונים כבר נמצאים בו. הנקודה ההיסטורית שהוצגה: פעם היה IDE, ולידו חלון צ'אט נפרד שאליו העתקנו קוד ושגיאות ידנית. "היינו הרבה פעמים עושים copy paste נותנים עוד קצת קונטקסט, שולחים לו את השגיאה" [1:30]. הדור הנוכחי איחד את השניים.

שכבת ה-Agentic IDE & Harness: מפתח אחד מול Kiro, Cursor, Claude Code ו-Codex

ההבדל אינו רק בנוחות: המוצרים האלה מביאים איתם את כלי קריאת וכתיבת הקבצים כחלק מהסביבה, כך שהמפתח לא מזין הקשר ידנית. "בואו ניקח את ה-ID הזה, נשים את הצ'אט בפנים" [1:30], ברוב המקרים על בסיס VS Code Open Source.

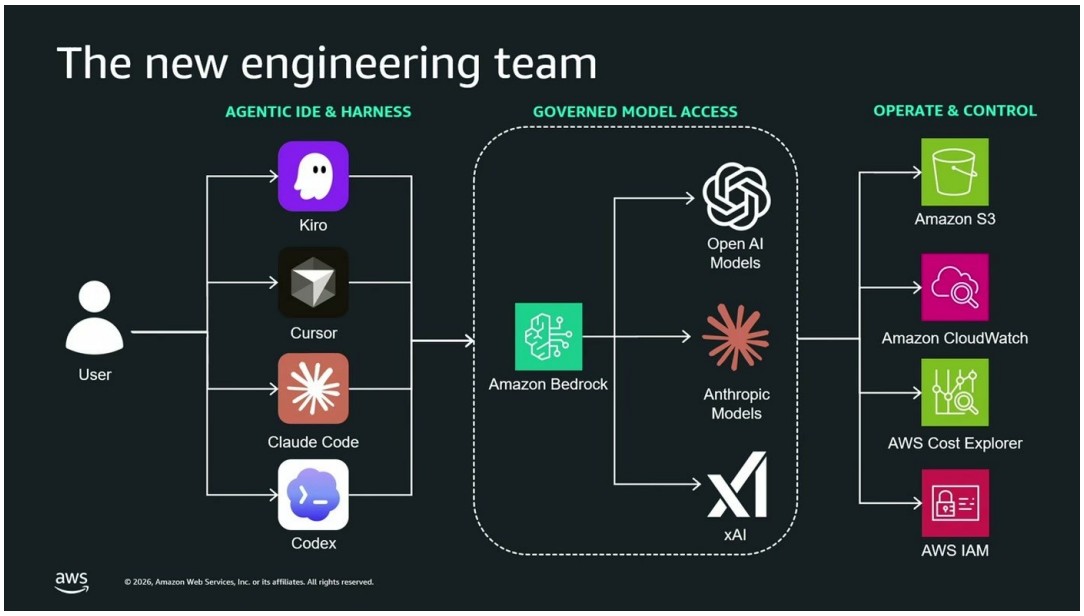
Kiro — ה-Agentic IDE של AWS

Kiro הוצג כמענה לבעיה של vibe coding שמתפרק בקנה מידה: "המטרה שלו העיקרית זה לאפשר לנו לעבור מפרוטוטייפ לפרודקשן" [3:01]. שלושת העמודים שעל הסלייד:

- **Spec-driven Development:** לא פותחים צ'אט ואומרים "יאללה בוא תעזור לי", אלא requirements → spec → פירוק ל-tasks, ו-Kiro מריץ sub-agents לביצוע המשימות [3:01].
- **Context Management:** זיכרון חוצה-סשנים של העדפות ושל ארכיטקטורת הפרויקט, כדי לא להזין את אותו הקשר בכל סשן חדש.
- **Persistency across devices:** כניסה מה-IDE, מהדפדפן או ממערכות אחרות אל אותה עבודה.

מתחת לזה: גישה מנוהלת למודלים

Amazon Bedrock הוצג כאן לא כשירות RAG אלא כשכבת הגישה למודלים: "בבדרוק אנחנו מסוגלים לגשת on-demand למודלים, של הפרווייידרים הכי טובים בשוק היום" — OpenAI, Anthropic, xAI ומודלי open weights [4:32]. הערך הארגוני הוא מה שנתלה מסביב.

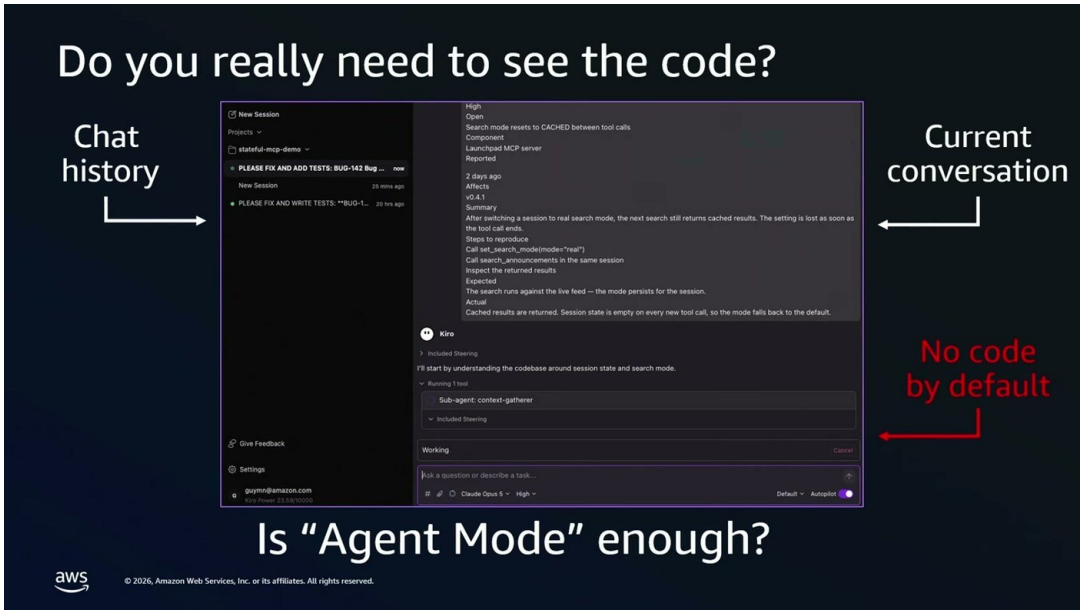


מימין לשמאל: ה-IDE, גישה ממושלת למודלים דרך Bedrock, ושכבת Operate & Control — S3 ל-invocation logs, CloudWatch, Cost Explorer ו-IAM

- **Invocation logs ל-S3:** "לדעת מי קרה למודלים בסיטואציה וכמה טוקנים זה עלה" [4:32].
- **CloudWatch ו-Cost Explorer:** מדדים, ושיוך עלות לרמת משתמש, סוכן או סשן — "אפילו לתייג את זה ברמת הסשן אם אנחנו רוצים לייצר סביבה מולטי טננט ולחייב את הלקוחות שלנו" [4:32].
- **IAM:** עבודה עם הרשאות במקום מפתחות שמסתובבים בין מפתחים.

הסדק הראשון: האם עוד רואים את הקוד?

כאן נשברת האינטואיציה של code review קלאסי. בדמו של תיקון קטן ב-CLI, התוצר היה שינוי שאיש לא קרא: "בום יש לנו שינוי של בין 30 ל-40 שורות שאנחנו בכלל לא יודעים מה קרה בהם אין לנו מושג" [6:02]. וזו כבר לא אנומליה של הטרמינל — גם ה-IDE-ים עצמם עברו למצב שבו הקוד אינו מוצג כברירת מחדל, מצב שכל מוצר קורא לו בשם אחר: agent mode, focus mode [6:02]. השאלה שעל הסלייד, "Is Agent Mode enough?", היא הגשר לחלק השני: אם ממילא לא קוראים כל שורה, השליטה חייבת לעבור מהעין של המפתח אל התשתית.



מסך שכולו שיחה: היסטוריה בצד אחד, השיחה הנוכחית בשני, ובאדום — No code by default

4. שלב שני — מכלי אישי לתשתית ארגונית

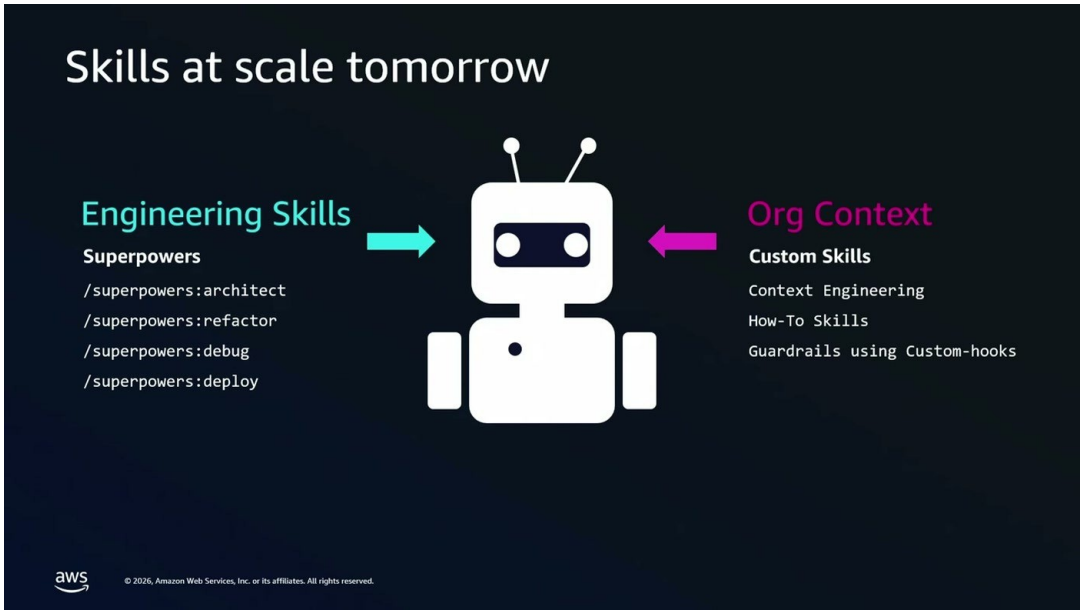
Alon Gendler לקח את החלק הזה, ופתח בתיאור איך AI מתפשט בפועל בארגון R&D: מפתח אחד מוריד skills, מפתח skills משלו, מוסיף MCP כדי להגיע ל-Salesforce או לבסיס נתונים, השכן רואה שזה עובד ומוריד גם הוא. "העניין הוא שזה לא יתפשט בצורה סדורה... לכל אחד יש את הכלים שלו" [9:07].

> **Alon Gendler [9:07] —** אנחנו למעשה רוצים לקחת את כל הכלים האלה ולהפוך אותם מקלי פרודקטיביטי אינדיבידואליים לממש תשתית ארגונית בטוחה

- **Security:** תחת אילו הרשאות רץ הסוכן ומי הוא בכלל — "זה הרבה יותר קל שאנחנו תחת איזה רול מסוים" — ומה בדיוק מגיע יחד עם ה-MCP או ה-skill שהורדנו [9:07].
- **Consistency:** האם אותו סוכן כותב כל פעם קוד באותו סגנון, "או שכל פעם אני אמצא את עצמי עם איזשהו קונקטור אחר למונגו דיבי" [9:07].
- **Visibility:** מה הסוכן עשה, באילו הרשאות — וגם עלות: "כולנו יודעים שיש עלויות מאוד גבוהות לטוקנים" [10:39].

האנלוגיה: מה הופך מפתח לפרודוקטיבי

המסגרת הרעיונית של החלק הזה היא פירוק של מפתח בכיר לשני מרכיבים. הראשון הוא Experience: עשר עד חמש-עשרה שנות ניסיון, יכולת לפרק דרישה למשימות, לצפות מקרי קצה, להכיר מתודולוגיה ולכתוב טסטים. השני הוא Org Context: חמש עד שבע שנים בארגון, ידיעה "באיזה API כדאי להשתמש, לאיזה קומפוננטה אפשר לעשות reuse". את שניהם צריך להעביר לסוכן — וכל אחד עובר בדרך אחרת [10:39].

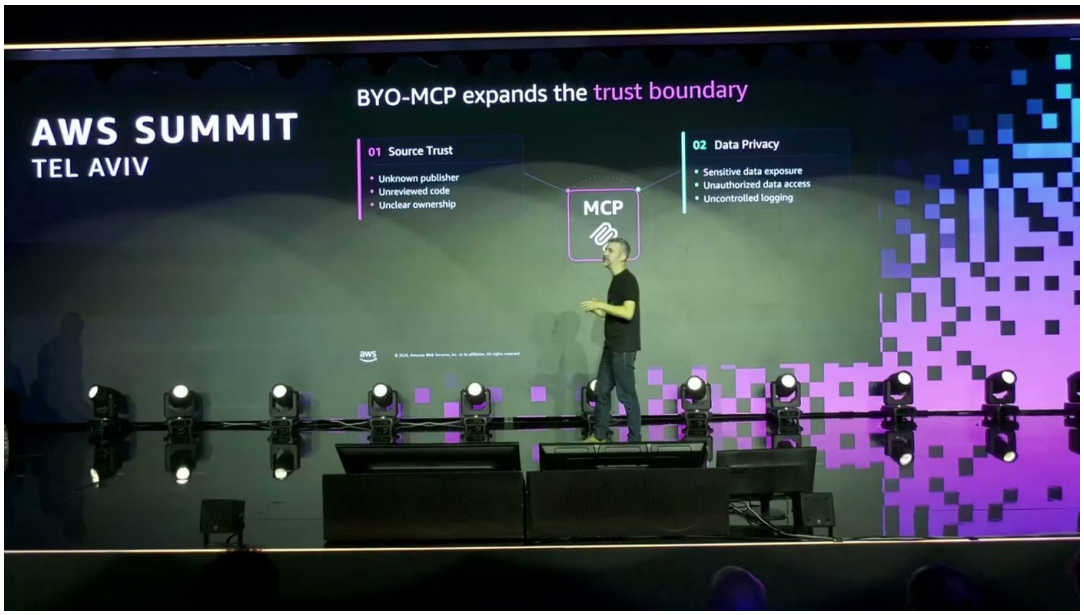


שני מקורות הידע של הסוכן: skills גנריים של הנדסה מצד אחד, ו-custom skills שנושאים את ההקשר הארגוני מצד שני

- **Superpowers — הצד הגנרי:** פריימוורק קוד פתוח. "סופר פאורס זה אופן סורס שאפשר להוריד בגיטאפ, הוא צבר תאוצה מאות אלפי סטארים בפחות מחצי שנה, ואם אתם הורידים אותו לארגון שלכם, הוא מביא 14 סקילים של מפתח טוב" [12:12] — ביניהם architect, refactor, debug ו-deploy.
- **Custom Skills — הצד הארגוני:** Context Engineering, שהוגדר כ"אבולוציה של prompt engineering" [12:12]: הארכיטקטורה, ה-API-ים הפנימיים, הסטאק וסכמת בסיס הנתונים.
- **How-To Skills:** "אם אתה רוצה להתחבר לדטאבייס איך אתה עשה את זה, אם אתה רוצה להתחבר לפרמיטיוס, לדטאדוג" — כדי שהסוכן לא ימציא את הגלגל בכל פעם [13:44].
- **Guardrails באמצעות custom hooks:** הוק שרץ רגע לפני הפעולה ושואל באיזו סביבה אנחנו. הרציונל נאמר במפורש: "כולנו שמענו על אג'נטים שמגיעים לי מסביבת הפיתוח, מסביבות נמוכות, מסביבות הפרודקשן וזה לא היה אמור לקרות" [13:44].

MCP: התועלת ומחיר האמון

MCP הוצג כשכבת התקשורת של הסוכן אל העולם — Jira, Slack, GitHub — וגם כדרך של צוותים פנימיים לחשוף יכולות: "יש לי צוות ריפורטינג, יש לי צוות ביזנס, יש לי צוות דבובס... אנחנו רוצים לחשוף MCPs ככה שאיג'נטים יוכלו לעבוד בתוך ה-R&D שלי" [15:18]. אלא שכל MCP שמתווסף מרחיב את גבול האמון.

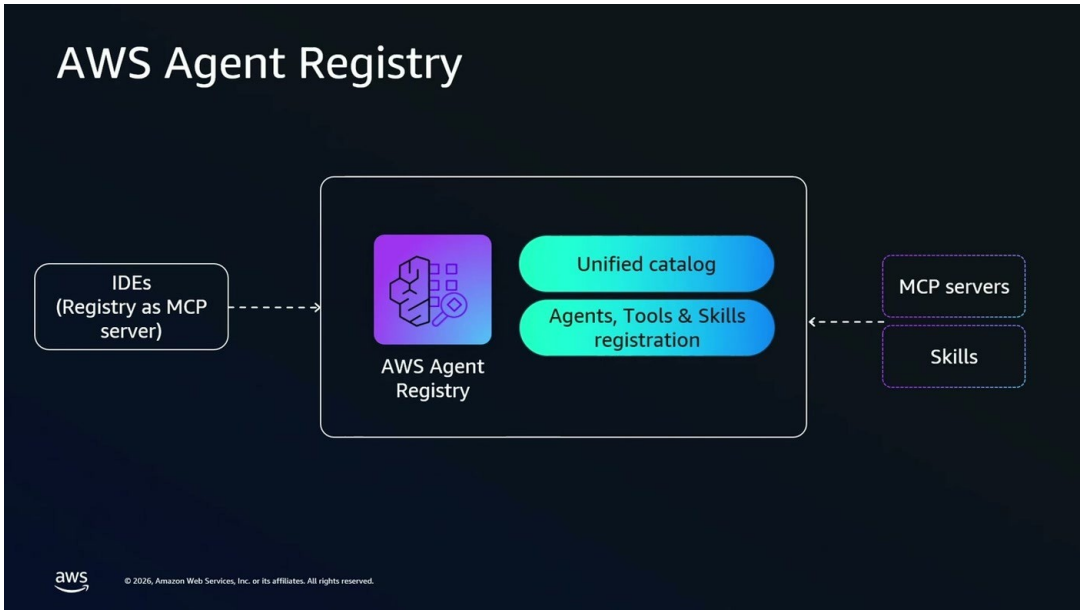


שני צירי הסיכון: Source Trust — מפרסם לא ידוע, קוד לא נסקר, בעלות לא ברורה; ו-Data Privacy — חשיפת מידע רגיש וגישה או לוגים לא מבוקרים

הדוגמה המוחשית שניתנה: מפתח שמצא MCP שמקטין את גודל הפרומפט וחוסך כסף, "אבל באותו אתגר גם הפרומטים שלי התחילו לצאת החוצה" [15:18]. ועל כך נוסף שכל skill או MCP מגיע לא פעם עם ספריות קוד — "והאם הספריות האלה הן ספריות בטוחות שאני רוצה אותן בארגון שלי" [16:48].

מ-allow-list לקטלוג

הצעד הראשון הוא allow-list ו-deny-list, שקיימים בכל הפלטפורמות הפופולריות; ב-Kiro זה MCP Registry, קובץ JSON מנוהל שנמשך מ-GitHub [16:48]. הגישה עובדת לעשרה או עשרים MCPs ונשברת אחר כך משתי סיבות: רשימה ארוכה "יכול מאוד לבלבל את האג'נט שלי... שגם עולה כסף מבחינת הפרומטים והטוקנים", ובארגון גדול יש MCPs מתחרים על אותה משימה [18:19].

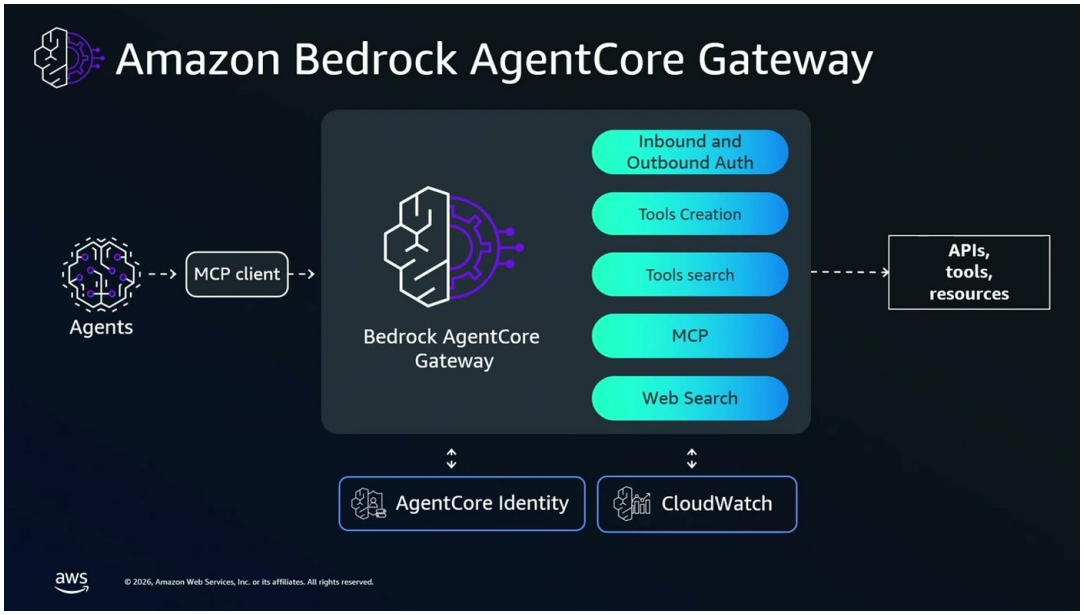


קטלוג אחוד לכל הארגון: רישום של agents, tools ו-skills, וה-registry עצמו נחשף ל-IDE כשרת MCP

AWS Agent Registry תואר כ"קטלוג אחד בטוח לכל הארגון שלי, שהוא יכול להנגיש לי גם MCPs וגם סקילס" [18:19]. התכונה שהודגשה כקריטית היא חיפוש סמנטי: "אם למעשה האג'נט שלי עכשיו רוצה לחפש MCP של בילינג וקוראים לו פיימנטס, אז הסמנטיק סרד של האג'נט רג'יסטרי יעזור לי למסוע את ה-MCP הנכון" [18:19].

AgentCore Gateway: זהות וטוקנים

הקטלוג פותר את שאלת הכלים הקיימים, אך לא את שאלת זהות הסוכן וההרשאות שלו. לשם כך הוצג Amazon Bedrock AgentCore Gateway, "שהמטרה שלו היא להיות סינגל פוינט לכל האייג'נטיק טראפיק בארגון שלי" [18:19]. סוכן — בין אם הוא רץ בתוך IDE ובין אם בשכבת compute — מגיע לגייטוויי, ומזדהה שם מול SSO כמו Okta או AWS SSO [19:54].



נקודת כניסה אחת: אימות נכנס ויוצא, יצירת כלים, חיפוש כלים, MCP ו-Web Search — עם AgentCore Identity ו-CloudWatch מתחת

- **ניהול טוקנים:** "איזה אגנט יכול לקבל את הטוקן לסלק, אז למעשה הגייטווי יעזור לי, ידאג לתת לו את ה-JWT טוקן, ידאג גם לרפרש את הטוקן" [19:54].
- **Tool Creation:** הפיכת API Gateway או Lambda קיימים ל-MCP בלחיצה, "כך שאני לא צריך להתחיל לממש את הפרוטוקולים בעצמי" [19:54].
- **Tool Search ו-Web Search:** חיפוש סמנטי על היעדים של הגייטווי, וחיפוש אינטרנטי לקבלת מידע עדכני מעבר למה שהמודל אומן עליו [19:54].
- **CloudWatch:** ויזיביליות לכל מה שעובר דרך הנקודה היחידה הזו.

הסיכום של החלק, כתרחיש אחד: "אני מבקש לפתור איזה Jira טיקט, אני מתחבר ל-MCP של Jira, ולכם מכן ל-MCP של GitHub... אני כותב קוד באמצעות הסופר פאור סקיל שלי בצורה מאוד מאוד טובה ואני בסוף מקבל פי-אר" [21:26].

5. שלב שלישי — סוכנים אוטונומיים

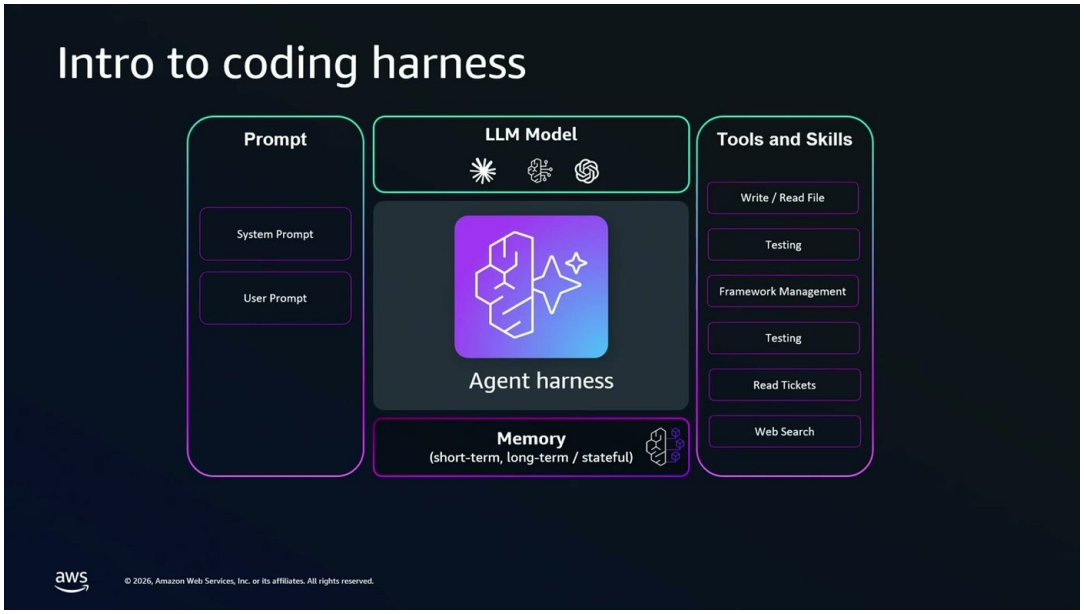
גם אחרי שאספנו את ה-MCPs, כתבנו skills ובנינו אמון — האדם עדיין מפעיל את התהליך, ממתין לו ובודק את התוצר. שתי הבעיות שהוצגו: Manual Triggering ו-Quality Inconsistency.

> **Guy Menahem [21:26] —** אנחנו מסתמכים על אנשים בתהליך, הם לא עובדים בלילה צריכים ללכת לישון מדי פעם, בסופי שבוע הם לוקחים חופש

והצד השני של אותה בעיה: "מה שמבחינתי הוא מספיק איכותי או יכול להיות ממש לא טוב למישהו אחר. יכול להיות שאני בחרתי מודל טוב לבצע את המשימה... ויכול להיות שמישהו אחר בחר מודל אחר" [21:26]. המסקנה נאמרת ישירות: "כדי להגיע באמת באמת לאפישינסים משמעותיים ולמכפילים נבואיים, אנחנו צריכים להשתמש באגנטים שהם לא מבוססים על בני אדם" [22:59].

ה-harness

"harness זה כמו תבנית. זה עוזר לנו לייצר אייג'נטים, הופעים של אייג'נטים, בתצורה שאנחנו רוצים" [22:59]. במקום סוכן שנוצר מחדש בכל שיחה, מקבעים מראש חמישה מרכיבים.



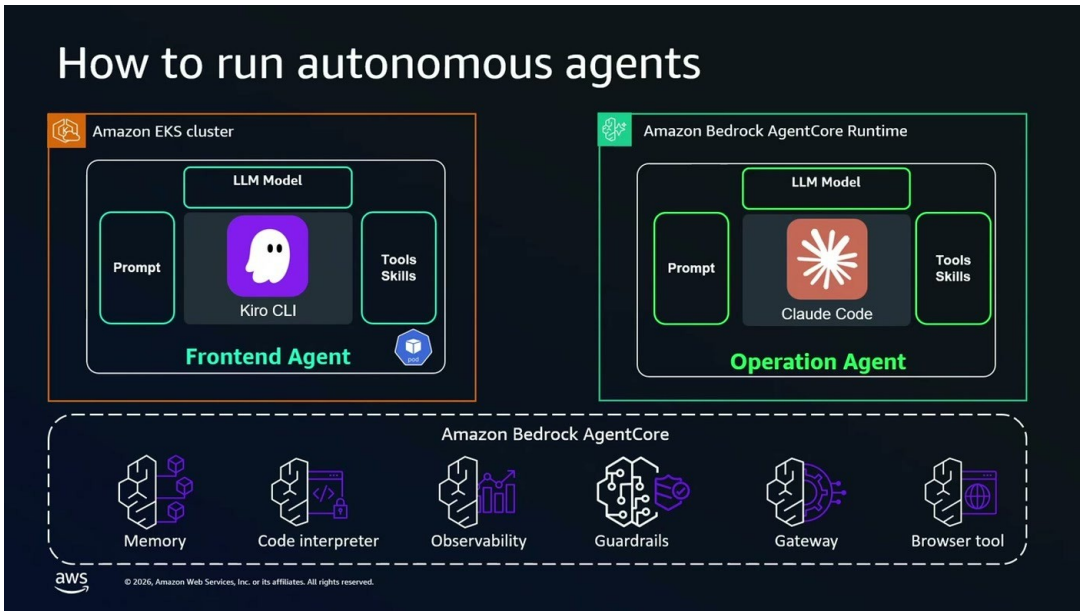
מרכיבי התבנית: system prompt ו-user prompt, המודל, ערכת הכלים וה-skills, וזיכרון קצר או ארוך טווח

- **System prompt:** תפקיד הסוכן וההקשר הקבוע שלו; ה-user prompt נשאר הדינמי, לכל משימה.
- **קיבוע המודל:** "לפעמים אנחנו נרצה מודל מאוד מאוד חזק, כדי לבצע משימה מאוד מאוד מורכבת, ולפעמים נרצה מודל חלש. הדבר המשמעותי זה שנדע איזה מודל" — וזו גם נקודת האופטימיזציה לעלות [22:59].
- **MCPs ו-skills מחוברים מראש:** "כדי לעצר קונסיסטנטיות בתוצאה" [22:59].
- **Memory:** שימור סשנים והמשך ריצות שכבר התחילו [24:32].

המימוש בפועל עובר דרך ה-CLI או ה-SDK של אותם כלים עצמם — Claude Code, Cursor, Codex ו-Kiro. CLI" בעצם מאפשר לנו הרצה יותר פשוטה, יותר נאיבית. SDK מאפשר לנו אורכסטרציה טיפה יותר מורכבת", והסשן התמקד ב-CLI [24:32]. ההגדרה הקריטית היא ההרצה במצב non-interactive: "תרוץ לבד, תספר לי בסוף איך זה היה", עם רשימת כלים ו-skills שמותר לו לגשת אליהם בלי אישור [24:32].

מכאן נגזרות תבניות מתמחות — סוכן frontend, סוכן backend, סוכן אינטגרציה, סוכן אופרציה — "כל דבר כזה יהיה תבנית שמתמחה במשהו מאוד מאוד ספציפי" [26:02]. הטריגר יכול להיות הודעה ב-Slack, טיקט מתויג ב-Jira או ב-Linear, או פתיחת PR.

איפה זה רץ



שתי אפשרויות הרצה זו לצד זו — pod ב-EKS מול runtime מנוהל — ומתחת לשתיהן אותה ערכת שירותי AgentCore

- **Amazon EKS:** "מי שלרוב יש לו EKS, וצוות שתומך ויודע איך עובדים עם המוצר, לרוב בוחר באופציה הזאת" [26:02].
- **Bedrock AgentCore Runtime:** סביבת ריצה מנוהלת — "אנחנו נותנים לו ממש דוקר אימג' והוא ממש אחראי להריץ את האייג'נט עבורנו", עם תמיכה בסשנים והמשכם [27:34].
- **Memory, Code Interpreter:** זיכרון ארוך-טווח והפקת עובדות ממנו; והרצת קוד בסביבת sandbox "לרוב מתאים שאנחנו מגמריטים קוד לא בטוח ורוצים שהוא לא יערוץ אצלנו בסביבה" [27:34].
- **Observability, Guardrails, Gateway, Browser tool:** מדדים ובסיס להרצת evaluations; סינון קלט ופלט; הגייטוויי מהפרק הקודם; ודפדפן לפעולות מול מערכות legacy שאין להן MCP [27:34].

6. המקרה של Reco: צי סוכנים בפרודקשן

Gilad Wisney, VP R&D ב-Reco, הביא את החלק היישומי. Reco היא חברת agentic security שמחברת את אפליקציות ה-SaaS של הלקוח, שואבת דאטה דרך API ומייצרת ממנו תובנות: discovery והרשאות, deep dive על posture management, הסוכנים עצמם, ולבסוף threat detection — "בין אם זה פעולות עצמם ברמת האיבנט או ממש ברמת הפרומפט" [29:04, 30:35].

הכאב שהוביל אותם לסוכנים הוא כאב תחזוקה, לא כאב פיתוח. "אנחנו היום תומכים במשהו כמו 250 אינטגרציות", והאפליקציות האלה משנות API-ים ומוסיפות מושגים והרשאות כל הזמן [30:35].

> **Gilad Wisney, VP R&D, Reco [30:35] —** אנחנו כבר מגיעים למצב שהצוואר בקבוק אצלנו של אנשים שאמורים לתחזק את הדברים האלה, כבר פשוט לא אוברסקל ואנחנו צריכים פתרון אחר

שלושת השלבים, מנקודת מבט של מנהל

המסע שלהם שוחזר בדיוק לפי שלושת השלבים של הסשן, וכל מעבר נחסם על ידי מכשול אחר. המעבר הראשון היה בכלל לא טכני:

- **מ-AI-assisted לסוכנים שמבצעים — חומה פסיכולוגית:** "הדבר הזה אני תפסתי אותו אצלי כמנהל כיותר שינוי פסיכולוגי אצל העובדים, שהם צריכים איכשהו להבין שהם קצת מאבדים שליטה על מה שהם עושים, מאבדים שליטה על הקוד" [30:35].
- **המכשול הבא — פער ידע:** "אנחנו בריקו עובדים כמונוריפו ויש לנו תיקיית סקילס מלאה והתחלנו לבנות אותה ולתחזק אותה ולשתף אותה בין כולם" [32:09].
- **המכשול האחרון — אורקסטרציה ותשתית:** המעבר לסוכנים שעובדים מקצה לקצה "ממש בלי מפתח בדרך" נתקע דווקא בתשתיות, וזה מה שהם בנו [32:09].

הפלטפורמה: סוכן הוא קובץ YAML

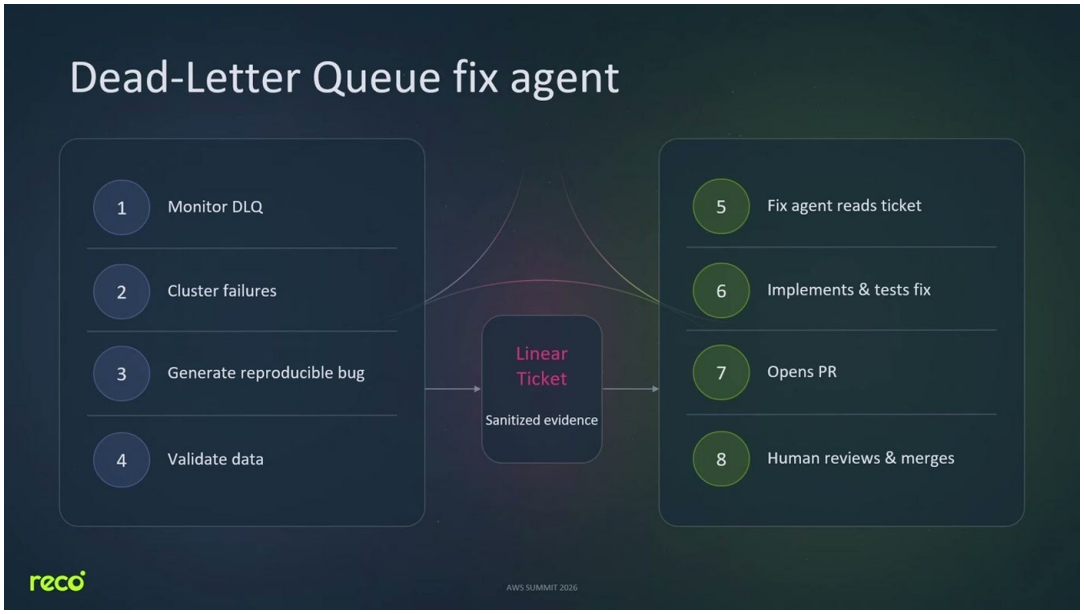
ברגע שמריצים צי סוכנים, "אנחנו כבר לא יכולים כל אחד בא וכותב מה שהוא רוצה" [33:42]. Reco בנתה שכבת תשתית שרצה כ-pods על EKS מול Bedrock — משם מגיעים גם שירות ה-LLM וגם ה-guardrails — והיא מספקת למפתח scheduling, caching, לוגים, observability ועלות כברירת מחדל.

הגדרת סוכן ה-DLQ: system prompt, מודל מתוך רשימה מאושרת, שאילתות SQL כמקור ידע, תלות בסוכן אחר, ואפילוג שפותח טיקט

"והיום כשמישהו בא לכתוב agent בריקו, הוא בעצם כותב ימל" [33:42]. שדות הקובץ שתוארו: system prompt; המודל, "מתוך סט מודלים שאנחנו מאפשרים"; SQL queries — "איזה knowledge אני רוצה לתת לו להריץ"; dependencies — האם הסוכן תלוי בסוכן אחר שירוץ לפניו; ואפילוג — מה עושים עם התוצאה, "זה יכול להיות לפתוח טיקט זה יכול להיות לייצר איזשהו ג'ייסון ולשלוח" [33:42].

ההפרדה שמאפשרת את הכל: פרודקשן מול פיתוח

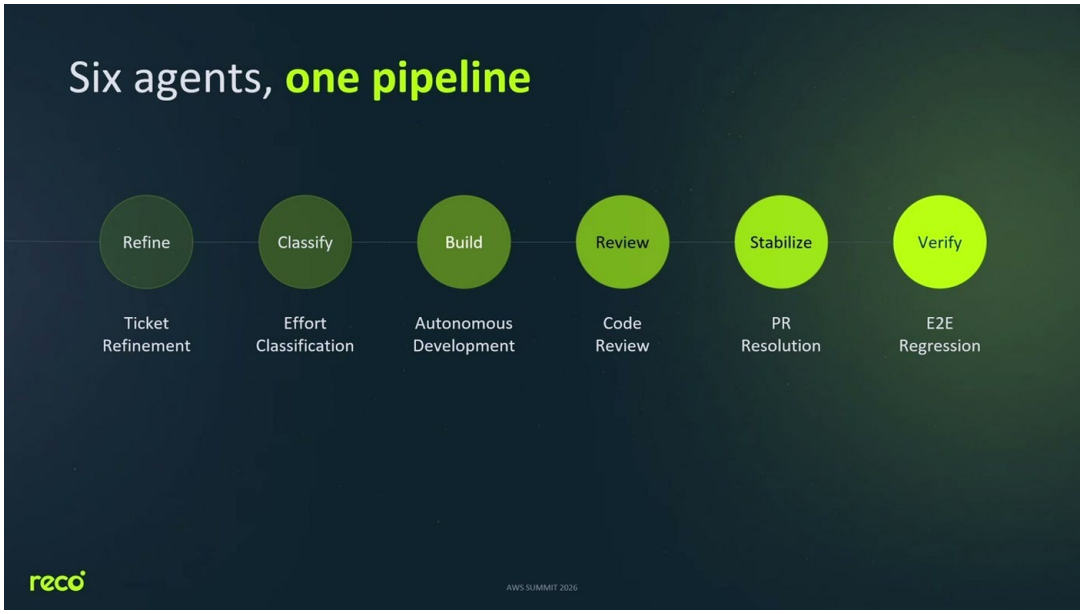
זו הנקודה הארכיטקטונית החשובה ביותר בחלק הזה. הסוכנים מחולקים לשתי אוכלוסיות עם מדיניות הפוכה. בסביבת הפרודקשן של הלקוח: "אלה אייג'נטים שהם מאוד מאוד אייסולייטד אין להם גישה לכלום כי בסוף מדובר פה על סביבה של לקוח". בסביבת הפיתוח: "יש להם גישה פתוחה יותר הם ניגשים לגיטב, הם ניגשים לאינטרנט" ולכל ה-MCPs וה-skills [35:12].



הטיקט ב-Linear הוא הגשר היחיד בין שתי הסביבות, והראיות עוברות בו רק אחרי sanitization

סוכן ה-Dead-Letter Queue קורא את טבלת השגיאות, עושה clustering לשגיאות חוזרות, מייצר מהן reproducible bug, ואז עובר שלב ולידציה: "אנחנו מבדים שאנחנו בתוך הבג שאנחנו הולכים לפתוח אין שום דאטה של לקוח או שום דבר שלא יכול לצאת החוצה מסביבת פרודקשן" [35:12]. ואפילו אז, הסוכן לא נוגע במערכת הטיקטים: "לאג'נט עצמו אין גישה למערכת טיקטינג, אצלנו ליניאר, יש ממש סקריפט שהוא מריץ ופותח טיקט" [35:12].

שישה סוכנים, pipeline אחד



מהטיקט ועד הרגרסיה: כל שלב הוא סוכן נפרד עם תפקיד אחד

"בשביל באמת לעשות איזשהו תהליך, פיתוח שהוא אפקטיבי ושעובד ושאנחנו יכולים לסמוך עליו, הוא מורכב אצלנו מפייפלן של שישה אגנטים" [36:45].

שלב	מה הסוכן עושה
Ticket Refinement	בונה acceptance criteria ופירוט, עובד על ה-Linear ומפצל ל-sub-tickets בעת הצורך
Effort Classification	מדרג את מורכבות הטיקט בסולם 1–5; כיום הסוכנים מטפלים ב-1 ו-2, ומתחילים להרחיב ל-3
Autonomous Development	מפתח ופותח את ה-pull request
Code Review	מורכב ממומחים נפרדים — ארכיטקטורה, best practices של Reco, התאמה בין ה-PR לטיקט; "משהו כמו עשרים מומחים"
PR Resolution	פותר את ההערות שעלו מה-review
E2E Regression	בונה סביבה ומריץ בדיקות, בין היתר ב-Playwright, עד לאישור שהטיקט מוכן

האדם נשאר בנקודה אחת בלבד: "בסוף אצלנו יש בן אדם שצריך לפתוח את הפיאר, להסתכל עליו, לראות שכל הטסטים עוברים וללחוץ בסוף על כפתור המרג'" [36:45]. סוגי הסוכנים האחרים שהוזכרו כרצים בפרודקשן: ולידציה של posture ושל אלרטים, זיהוי אנומליות בדאטה, ובדיקה שכל מה שנורמל בדאטה אכן מוצג ללקוח [32:09].

7. מה לוקחים מכאן

שלוש ההמלצות שניתנו בסיום, ומעליהן שתי מסקנות שעולות מהסשן כולו.

- **להתחיל קטן ומתוחם.** "הראשון שאנחנו רואים שעובד זה פשוט להתחיל בקצר" — לדוגמה סוכן שרואה טיקט חדש ב-Linear ומחליט אם הוא בגודל שלו: "כל עוד זה יהיה scoped, זה אחלה התחלה" [38:20].
- **Packaging.** פונקציות שחוזרות על עצמן בין סוכנים — לארוז כ-skill או כ-MCP במקום לשכפל [38:20].
- **זהות והרשאות לסוכן.** שיהיה ברור מי הסוכן ומה הוא הולך לעשות, ושלא יקבל יותר מדי טוקנים — "אלא שהאג'נט קור גייטוי לדוגמה ינהל את זה" [38:20, 39:51].
- **ההשקעה האמיתית היא בהקשר הארגוני, לא בפרומפט.** שני החלקים הארוכים בסשן — custom skills ו-MCP governance — עוסקים בהעברת ה-Org Context לסוכן. זה גם מה ש-Reco זיהתה כפער המרכזי שלה.
- **גבול הגזרה הוא ארכיטקטוני, לא הוראה בפרומפט.** בדוגמה של Reco הסוכן בסביבת הלקוח פשוט לא מחובר לשום דבר, ופתיחת הטיקט נעשית בסקריפט חיצוני. זו ההגנה שעובדת גם כשהמודל טועה.

המשפט שסגר את החלק של Reco

שאלות פתוחות להמשך

נושא	למה זה חשוב	חותמת זמן
מדידת איכות התוצר	לא הוצגו שיעורי הצלחה, אחוז PR-ים שנדחו או עלות לטיקט	—
Evaluations על סוכנים	הוזכר כיכולת של AgentCore, בלי דוגמה למדדים בפועל	27:34
בחירת מודל לכל תפקיד	נאמר שזו נקודת האופטימיזציה לעלות, בלי קריטריון	22:59
בשלות AWS Agent Registry	הוצג כשירות; לא נאמר מה מצב הזמינות שלו	18:19
הרחבה מעבר למורכבות 2	Reco מרחיבה ל-3; לא נאמר מה חוסם מעבר לכך	36:45

8. מילון מונחים

מונח	משמעות כפי שהוצגה בסשן
Agentic IDE	סביבת פיתוח שבה הצ'אט והכלים לקריאה וכתיבה של קבצים הם חלק מהמוצר, לרוב על בסיס VS Code Open Source
Kiro	ה-Agentic IDE של AWS, שמכוון למעבר מפרוטוטייפ לפרודקשן
Spec-driven Development	תהליך מסודר של requirements → spec → פירוק ל-tasks, במקום צ'אט חופשי
Harness	תבנית מקובעת ליצירת מופעי סוכן: system prompt, מודל, כלים, skills וזיכרון
Non-interactive mode	הרצת סוכן ב-CLI בלי המתנה לאישור או לתשובה מהמשתמש
Skill	יחידת ידע או נוהל שמוזנת לסוכן; גנרית (Superpowers) או ארגונית (custom skill)
Superpowers	פריימוורק קוד פתוח שמביא לפי המוצג 14 skills של מפתח טוב — architect, refactor, debug, deploy
Context Engineering	הזנת ההקשר הארגוני לסוכן — ארכיטקטורה, API-ים, סטאק וסכמת בסיס הנתונים
Custom hooks	הוק שרץ לפני פעולה של הסוכן ומאפשר לעצור אותה, למשל לפי סביבה
MCP	פרוטוקול אחיד שדרכו סוכן מתקשר עם API-ים, מערכות וסוכנים אחרים
AWS Agent Registry	קטלוג ארגוני אחד ל-MCPs ול-skills, עם רישום וחיפוש סמנטי
Bedrock AgentCore Gateway	נקודת כניסה יחידה לתעבורה אג'נטית: זיהוי הסוכן, ניהול טוקנים, יצירת כלים וחיפושם
Bedrock AgentCore Runtime	סביבת ריצה מנוהלת לסוכנים, מקבלת Docker image ותומכת בסשנים
Dead-Letter Queue agent	הסוכן של Reco שקורא שגיאות מטבלה, מקבץ אותן ומייצר מהן באג משוחזר