# להריץ AI בסקייל: אופטימיזציה, Observability ו-Evaluation

AWS Summit Tel Aviv 2026 ,סיכום סשן — TLV-AIM302

מור אלדר, Senior AI Specialist, AWS | ארז ויינשטיין, Solutions Architect, Startups, AWS | נעמה דמתי, VP Research, CX Division, NICE

10 בספטמבר 2026 | משך: 38 דקות | הופק מהקלטת הסשן

סיכום מאת קובי אלמוג

# על המסמך הזה

המסמך מסכם את הסשן TLV-AIM302 מתוך AWS Summit Tel Aviv 2026, במסלול Building AI and Agentic Applications. הוא נבנה מתמלול אוטומטי של ההקלטה המלאה ומהסליידים שחולצו מהווידאו, כך שאפשר לחזור לתוכן בלי לצפות שוב בכל שלושים ושמונה הדקות. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

הסשן בנוי משלושה חלקים ברצף: מור אלדר על אופטימיזציה, ארז ויינשטיין על Observability ועל Evaluation, ונעמה דמתי מ-NICE על המימוש בפרודקשן אצל לקוחותיה.

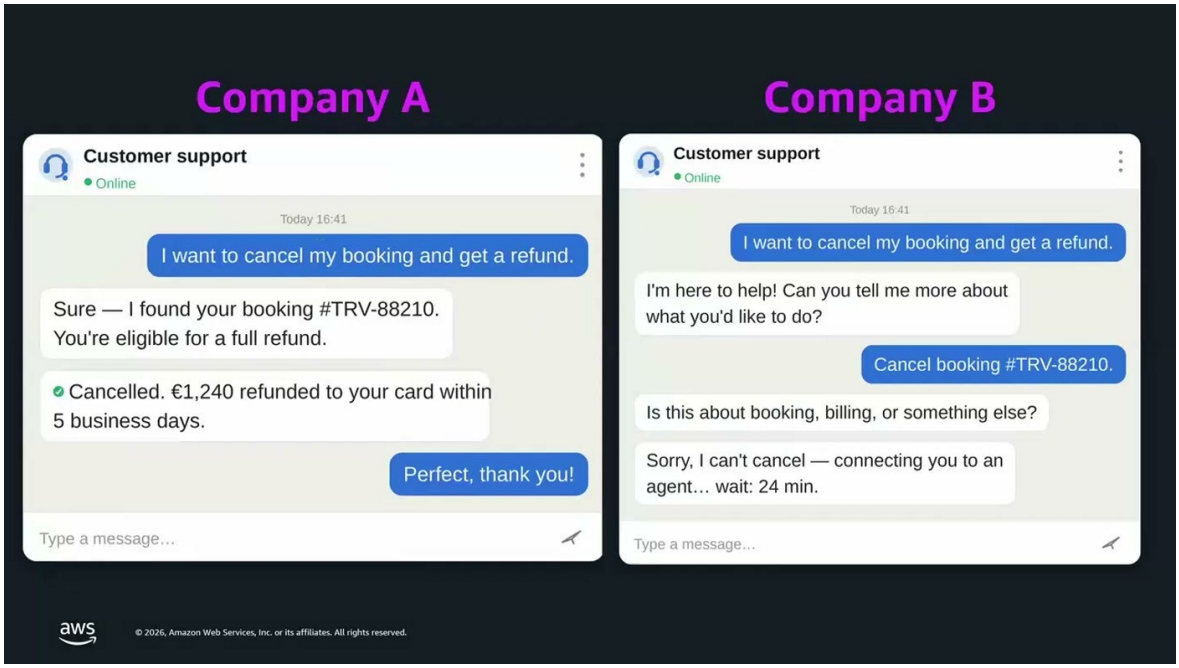
### איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה נאמר, ומה ישים בארגון גדול.
- **פרק 2 — אופטימיזציה:** המשולש איכות/מהירות/עלות, ושלוש דוגמאות מעשיות — מודלים, קונטקסט, Inference.
- **פרקים 3–4 — Observability:** למה הכלים הקלאסיים לא מספיקים, ארבע שיטות ל-Cost Attribution, ואנטומיה של ריצת agent.
- **פרק 5 — Evaluation:** מה נמדד, מתי, ובאילו evaluators.
- **פרק 6 — המקרה של NICE:** שלוש שכבות מטריקות, למה נבחר LLM Gateway, ואיך זה נסגר להמלצות.
- **פרק 7 — מה לוקחים מכאן:** מסקנות ונקודות להמשך.

**הערת מקור** המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול טוב בקטעים המקצועיים, אך מונחים לועזיים ושמות מוצרים הופיעו בשיבוש פונטי (למשל "אג'נט קור" = AgentCore, "בדרוק" = Bedrock) ותוקנו ידנית מול הסליידים. הציטוטים אומתו מול ההקלטה בחותמת הזמן המצוינת ומובאים כלשונם, כולל שיבושי תמלול קלים. מספרים שמופיעים על הסליידים בלבד צוינו ככאלה.

# 1. תקציר מנהלים

הסשן יצא מתצפית פשוטה: שתי חברות יכולות לבנות צ'אטבוט על אותם מודלים ואותם כלים, ולספק חוויה הפוכה לחלוטין. מור אלדר: "שתי החברות האלה נשנות על אותה טכנולוגיה, הזמינה לשתיהן אותם כלים, אותם מודלים איך יכול להיות שהחוויה שאנחנו מקבלים כל כך שונה?" [1:32]. התשובה שהוצעה לאורך כל הסשן היא שההבדל לא נמצא בטכנולוגיה אלא בתשתית שמסביבה — אופטימיזציה, Observability ו-Evaluation כלולאה אחת.



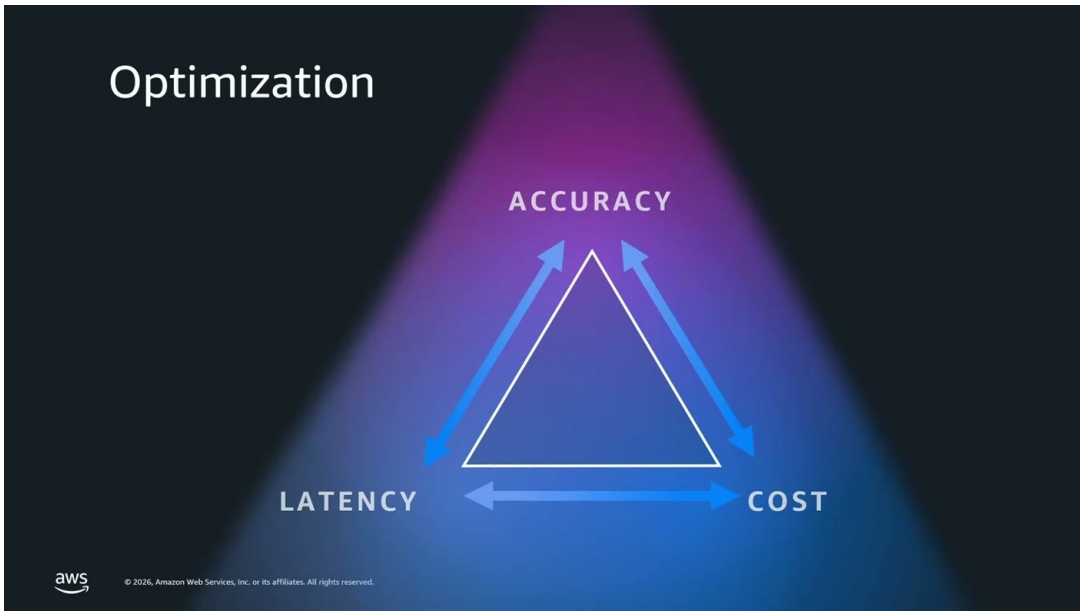

*שתי אותן בקשות ביטול, אותה טכנולוגיה — Company A סוגרת את הפנייה, Company B מגלגלת לנציג אנושי*

- **אין פרמטר אחד לאופטימיזציה — יש תיעדוף לפי משימה.** בהתכתבות שוטפת מהירות קודמת לאיכות; בבקשת ביטול, שבה עלות הטעות גבוהה, האיכות קודמת. "אותו מוצר, שתי משימות, פרמטרים שונים לתיעדוף" [3:03].
- **הרווחים הגדולים יושבים על מה שכבר קיים.** Prompt Caching הוצג כמניב עד 90% הוזלה ועד 85% שיפור ב-latency באותה איכות; Inference Tiers מציעים 50% הנחה למשימות שאינן רגישות לזמן תגובה — שינוי בקונפיגורציה, לא במודל [9:15].
- **כלי ה-Observability הקלאסיים לא מספיקים בעולם האג'נטי.** שלוש סיבות שהוצגו: חוסר דטרמיניזם, תהליך reasoning אטום, וכשלים שקטים — "בדרוק מחזיר לנו 200, תוקנים חזרו בצורה מוצלחת" בזמן שהתוכן עצמו שגוי [12:19].
- **ייחוס עלות הוא שאלת גרנולריות, לא שאלת כלי.** ארבע שיטות הוצגו בסדר עולה של רזולוציה: IAM Principal Attribution, Application Inference Profiles, Per-Request Metadata Tagging, ו-LLM Gateway — כל אחת עם מחיר תפעולי משלה.
- **Evaluation הוא מה שמפריד בין agent שרץ לבין agent שעובד.** AgentCore Evaluations מדרג טראג'קטוריה שלמה מדאטה פרודקשן, עם 13 evaluators מובנים ואפשרות ל-custom, בשני מודים: Online על דגימת תנועה חיה, ו-On-demand בתוך CI/CD [23:13].
- **NICE הוכיחה שהלולאה נסגרת בפועל.** הפלטפורמה לא מסתפקת בדשבורדים אלא מייצרת המלצות שהלקוח מאשר בלחיצת כפתור — "אנחנו לא עוצרים רק להראות מטריקות בדשבורד, אנחנו גם מנתחים את המידע הזה ומביאים המלצות לאופטימיזציה" [31:04].

### רלוונטיות לארגון גדול

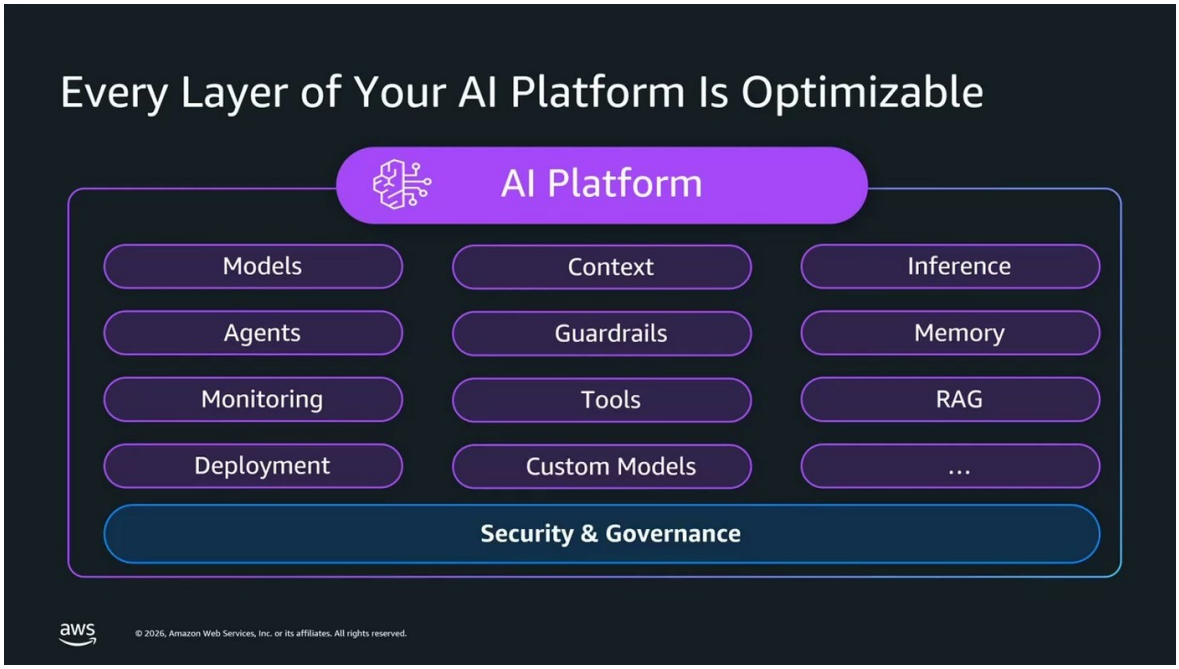
שלושת הנושאים שהוצגו הם בדיוק הפערים שמעכבים העלאת AI לפרודקשן בארגון מפוקח: מי צרך כמה ובשם מה (ייחוס עלות לפי יחידה עסקית או tenant), מה בפועל קרה בתוך הפנייה (trace מלא במקום שורת לוג), והאם התשובה עמדה במדיניות הארגונית (evaluator שבודק נאמנות למקור והסקלציה נכונה). המקרה של NICE — שבו agent הבטיח החזר כספי בסתירה למדיניות החברה — הוא בדיוק סוג הכשל שדשבורד ירוק לא יתפוס.

## 2. אופטימיזציה: איכות, מהירות, עלות




*שלושת הקדקודים שביניהם מתנהל התיעדוף — Accuracy, Latency, Cost*

מור אלדר פתחה בהצגת המשולש: "בעולם אידיאלי אנחנו נרצה לשפר את שלושתם" [1:32], אבל "במציאות אנחנו יודעים ששיפור של אחד לרוב יבוא על חשבון האחר המטרה שלנו היא למצוא את המקומות שבהם זה לא קורה" [3:03]. הדוגמה שליוותה את כל הסשן היא צ'אטבוט שירות לקוחות של חברת נסיעות — שתי משימות באותו מוצר, שני תיעדופים שונים: בהתכתבות שוטפת המהירות מכריעה, "כי לא משנה כמה תהיה איכותית התשובה, אם היא תגיע באיחור היא לא תהיה רלוונטית" [3:03]; בבקשת ביטול, משימה מורכבת יותר שבה "העלות לטעות גבוהה יותר", האיכות גוברת גם במחיר מהירות.



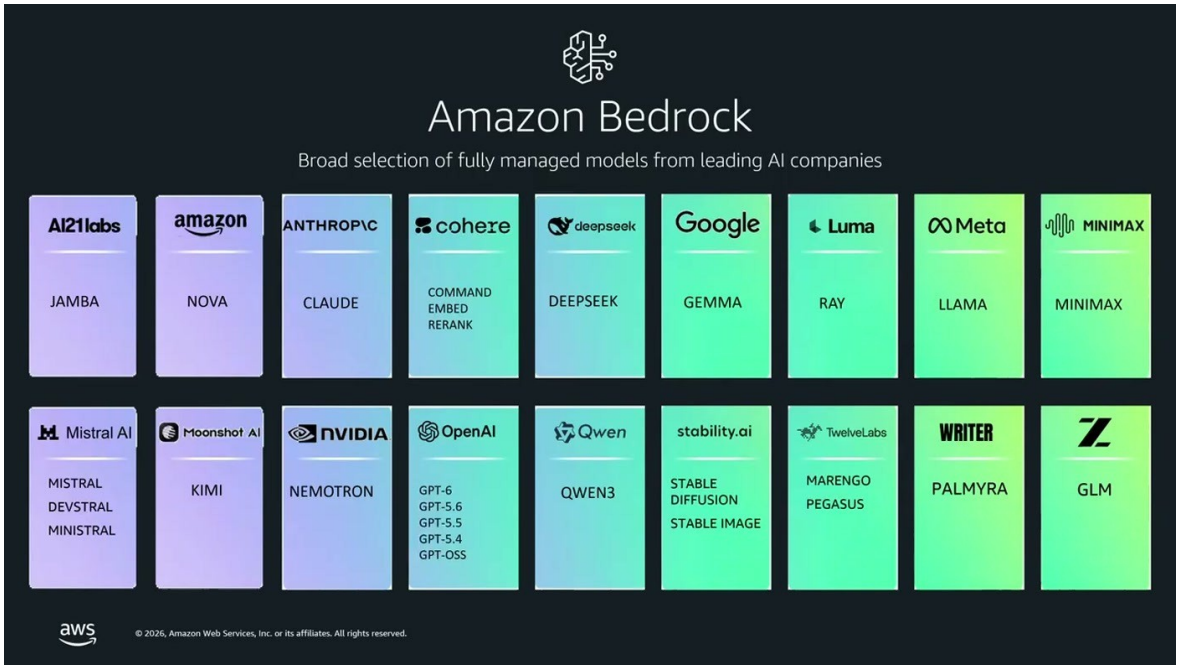

*מפת הרכיבים שניתנים לאופטימיזציה — מודלים, קונטקסט, Inference, Agents, Guardrails, Memory, Tools, RAG ועוד*

הנקודה המסגרתית: "מאחורי כל תשובה וקריאה של הצ'אטבוט שלנו יש מערכת שלמה המודל, הקונטקסט שנשלח אליו, הזיכרון, הסוכנים, הכלים שבהם הם משתמשים", וכל אחד מהם הוא בו-זמנית "שורה בחשבונית", זמן תגובה, וגורם איכות [3:03, 4:37]. מתוך כל אלה הוצגו שלוש דוגמאות מפורטות.

## 2.1 בחירת מודל לכל משימה

"ההמלצה הגורפת היא להתאים את המודל האידיאלי פר משימה" [4:37] — מודלים זולים ומהירים למשימות פשוטות, מודלים חזקים ויקרים למשימות מורכבות. פער העלות שצוין בין המודל הקטן למודל החזק באותה משפחה הוא עד פי 20. התצפית מהשטח: "אנחנו רואים לא מעט לקוחות שעדיין נשארים עם אותו מודל שהם איתו יצאו לדרך ולא משתמשים במודלים שונים פר משימה" [4:37].

בהמשך נעשה סדר בהגדרות, מפני שהמונחים מתערבבים: מודלים Proprietary הם מודלים סגורים ומסחריים הנצרכים רק דרך API מנוהל ובתשלום לפי צריכה; מודלים Open Source חושפים גם את הקוד וגם את הדאטה שעליו המודל אומן; ומודלים Open Weights — הקטגוריה שנמצאת היום במרכז — חושפים את המשקולות בלבד. עליהם נאמר שאפשר לעשות fine-tune, לצרוך אותם היכן שרוצים וגם דרך API, שייתכנו מגבלות רישוי לשימוש מסחרי או להפצה, ושהמחירים שלהם "משמעותית יותר זולים והם שיפרו משמעותית את הביצועים שלהם לאחרונה" [6:10].




*קטלוג הספקיות ב-Amazon Bedrock כפי שהוצג — מודלים סגורים לצד Open Weights באותה נקודת צריכה*

> **שימו לב** אזהרת ההשוואה, והיא אולי המסר המעשי ביותר בפרק: השוואת מודלים חייבת להיעשות ברמת המשימה השלמה ולא ברמת הטוקן. מודל יקר שמסיים באיטרציה אחת עשוי לצאת זול ומהיר יותר ממודל זול שנזקק לחמש. "ואם אנחנו נסתכל רק על המחיר פר טוקן או על הנתונים היבשים של השוואת המודלים אנחנו נבחר במודל הלא נכון" [7:41].

על Amazon Bedrock נאמר שזמינים בו למעלה מ-100 מודלים מהספקיות המובילות, ושהערך שלו בהקשר הזה הוא היותו "מקום אחד שבו אתם יכולים לצרוך את המודלים, להשוות בין המודלים ולהחליף מהר כשזה משתלם", בסביבה שבה המידע אינו משותף עם ספקיות המודלים [7:41].

## 2.2 אופטימיזציית קונטקסט: Prompt Caching

ההנחה שבבסיס: "כל מה שנשלח למודל נשלח מחדש בכל בקשה" [7:41]. ה-system prompt — קובץ שמגדיר את תפקיד הסוכן, מדיניות ושימוש — נשלח מילה במילה בכל קריאה. "אין סיבה שנשלם עליו בכל פעם וגם אין סיבה שנחכה לו בכל פעם". המנגנון פשוט: מסמנים בקריאה למודל היכן נגמר החלק הקבוע והיכן מתחיל המשתנה.


*נקודת ה-cachePoint בגוף הבקשה, ולצדה שתי המספרים שהוצגו כתוצאה*

"החלק הקבוע ישמר בקש ולא ישלח מחדש בכל בקשה והתוצאה עד 90% פחות בעלות ועד 85% פחות בלייטנצי באותה האיכות בדיוק" [9:15]. ההמלצה שנלוותה: "אם זה רלוונטי אצלכם ואתם לא מממשים את זה היום, זה מקום מעולה להתחיל בו".

## Inference Tiers 2.3

הדוגמה השלישית נוגעת לדרך ההרצה ולא לתוכן: בצ'אטבוט יש משימות real-time ויש משימות שהרגישות שלהן ל-latency נמוכה, "אין סיבה שנשלם על כולן אותו הדבר" [9:15]. Amazon Bedrock Inference Tiers מאפשר להתאים את שכבת ההרצה למשימה — "אותו המודל, אותה איכות, דרך הרצאה שונה". ארבע האפשרויות שהוצגו: Standard כברירת מחדל, Priority לזמני תגובה מהירים יותר בתוספת מחיר, ו-Flex ו-Batch שמעניקים 50% הנחה ממחיר ה-Standard בתמורה לזמני תגובה גמישים.

הפרק נסגר בשתי שאלות שמעבירות את הבמה: "על סמך איזה מידע אנחנו נחליט איפה להתחיל וברגע שהחלטנו איך אנחנו יודעים עם השינוי שאנחנו בוחנים באמת מיטיבים התוצאות" [9:15].

# 3. Observability: למה הכלים הקלאסיים לא מספיקים

ארז ויינשטיין פתח בהודאה שהמושג מוכר — טרייסים, לוגים, דשבורדים, אלרטים — "אבל בכל זאת משהו השתנה בעידן AI." [10:48]. את השינוי הוא הדגים בתרחיש: לקוח פונה ל-agent, מבקש ביטול סמוך למועד הטיסה, וה-agent מודיע לו שהוא זכאי להחזר מלא ומתחיל בטיפול. "אבל מדינות החברה שלנו אומרת בדיוק דבר הפוך שלפי המדיניות אין refund כאשר המועד הטיסה הוא קרוב ב-14 יום לפני". הלקוח מתקשר, זועם, מציג צילום מסך של התשובה, ואז מגיע החיפוש בלוגים.


*מה שנשאר בידי צוות התחקור אחרי הפנייה — רשומת INFO אחת עם session, status, latency ומספר טוקנים*

"שורת לוג אחת, זהו זה כל מה שיש לנו אנחנו לא יודעים איזה קריאות API אייג'ן שלנו הוציא אנחנו לא יודעים איזה קריאות, איזה למ הוא הוציא וכמה אנחנו לא יודעים איזה מסמכים הוא שלף מהנולדש בייס אין לנו מושג על איזה מידע הוא נשען" [12:19]. ומיד אחרי זה השאלה שאין עליה תשובה: האם זה מקרה בודד, או שזה קרה עוד הרבה פעמים.

### שלוש הסיבות שהוצגו

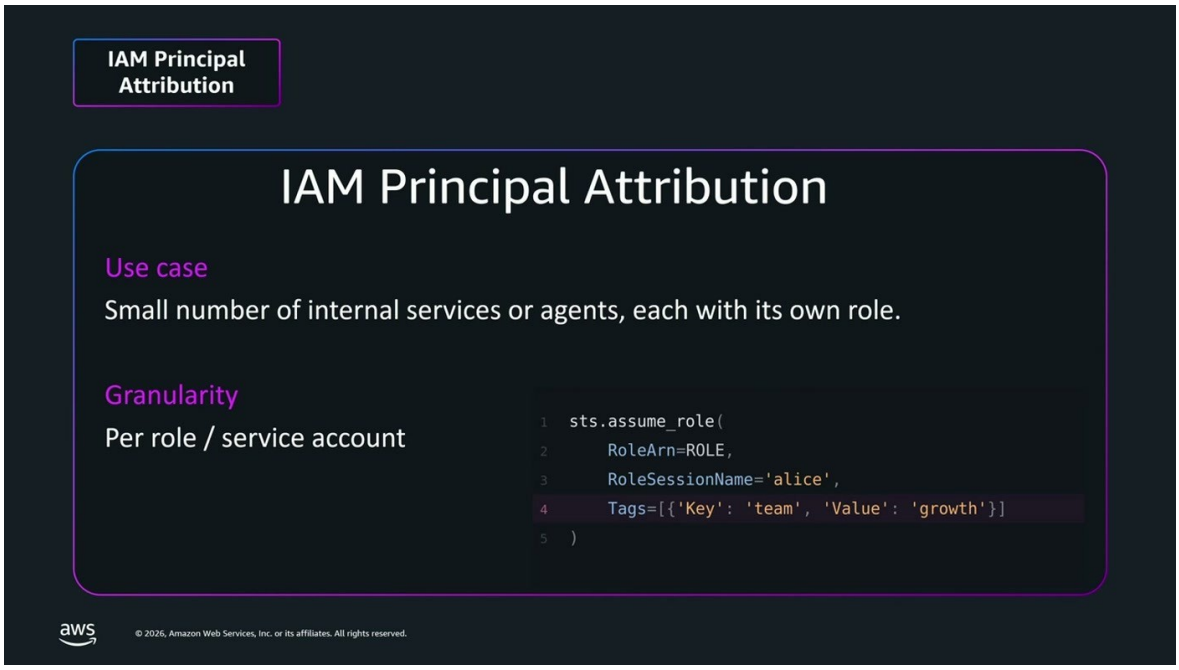
- **חוסר דטרמיניסטיות.** "אותו ה-input כאשר ניתן אותו לאג'נט אנחנו יכולים לקבל תשובות מעט שונות בחירת כלים אחרת" [12:19]. הגישה הקלאסית של שחזור הבאג פשוט לא תופסת.
- **תהליך reasoning אטום.** בעולם התוכנה הקלאסי יש if/else ואפשר להוסיף print ולוגים ולהבין את הלוגיקה; כאן זה לא קורה.
- **כשלים שקטים.** "אין לנו exceptions שנזרקים, אין לנו stack traces בדרוק מחזיר לנו 200, תוקנים חזרו בצורה מוצלחת" — והשגיאה יושבת בתוכן עצמו [12:19, 13:54].

### שלושה היבטים של Observability

מאותה פנייה נגזרו שלוש שאלות שונות זו מזו: בהיבט הפיננסי — כמה עלתה הפנייה (בדוגמה שהוצגה: 30 סנט); בהיבט האופרטיבי — מה בפועל קרה בתוכה, כמה קריאות ל-LLM ואילו tool calls; ובהיבט העסקי, שהוגדר כחשוב מכולם — "האם זה בכלל היה שווה את זה 30 סנט עבור טיפול של לקוח נשמע די טוב אבל אם נציג אנושי בכל זאת עלה אחר כך לשיחה וטיפל אז כנראה ששילמנו פעמיים ונרצה לדעת כמה פעמים זה קרה" [13:54].

# 4. ארבע שיטות לייחוס עלות, ואנטומיה של ריצת agent

השאלה "לאן הלכו הטוקנים שלי" נשענת, לדברי ארז, על שאלה מקדימה אחת: "מהי יחידת הפילוח שלכם? רמת הסרוויס, רמת האג'נט, ביזנס ליין, צוות מסוים, אולי טננט או שילוב של כמה דברים" [13:54]. ארבע השיטות שהוצגו מסודרות בסדר עולה של רזולוציה ושל מורכבות.



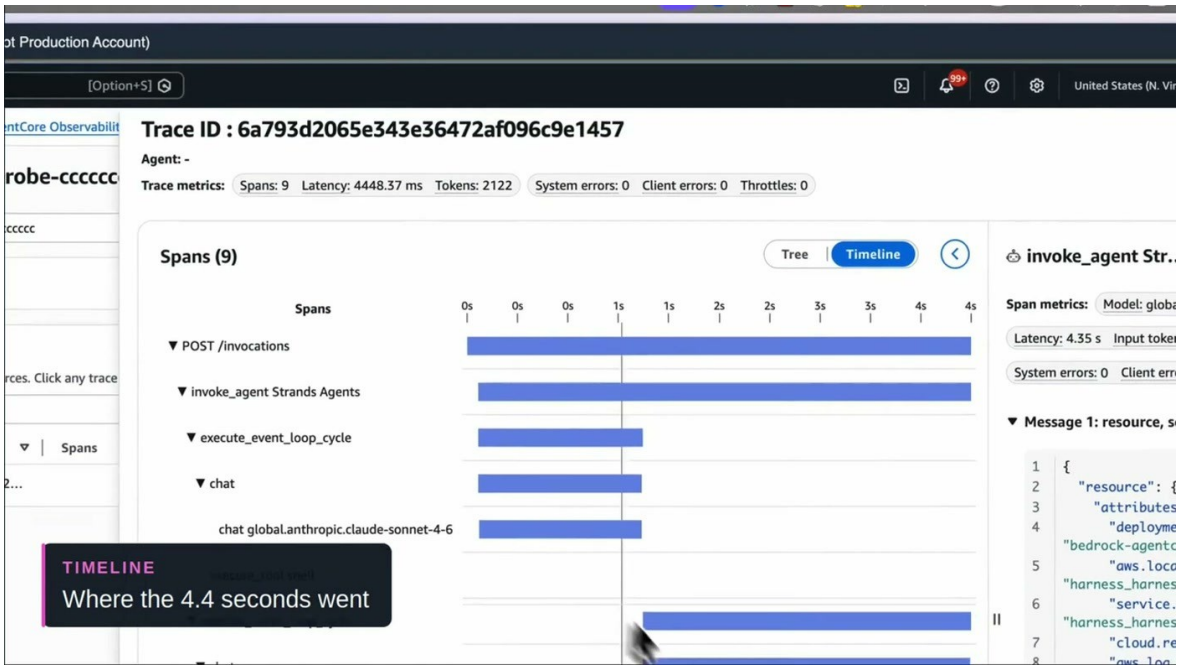

*השיטה הראשונה בפירוט — use case, רמת הגרנולריות, והפרמטרים ב-sts.assume_role*

- **IAM Principal Attribution.** נשענת על כך שה-agent פונה ל-Bedrock עם IAM Role ונושא איתו את הזהות; הוספת תגיות ל-AssumeRole מאפשרת פילוח גם לפי RoleSessionName וגם לפי התגיות [15:27].
- **Application Inference Profiles.** פתרון למצב ששני agents חולקים את אותו Role — "מצב שקורה בלא מעט ארגונים". במקום להעביר ל-agent model ID, מייצרים פרופיל שמצביע על מודל, מתייגים אותו ומעבירים ARN. התוצאה: שתי שורות נפרדות בחשבון [15:27].
- **Per-Request Metadata Tagging.** תגיות ברמת הקריאה הבודדת ל-Bedrock, שמאפשרות חיפוש ב-CloudWatch ומענה לשאלות ברמת ה-tenant — "איזה טננט בזבז הכי הרבה טוקנים אתמול באמצע הלילה" [15:27].
- **LLM Gateway.** רכיב ביניים בין ה-workloads לבין ספק ה-LLM שמרכז את כל הקריאות. מאפשר פילוח לפי agents, tenants וסוגי מודלים, ובנוסף שליטה: rate limits, budgeting, ואפילו "לעצור agent שמשתולל ושורף לי תוקנים בלי הכרה" [17:03].

> **מחיר הפתרון** ההסתייגות שנלוותה ל-Gateway: "אנחנו מוסיפים פה עוד רכיב במערכת שלנו שצריך לעשות לו סקייל, צריך לדוג לו ל-[17:03] "security — ובהמשך הפניה לאירוע האבטחה ב-LiteLLM כנקודה למחשבה. מי שבוחר בשיטה הזאת מקבל גרנולריות ושליטה, ולוקח על עצמו רכיב נוסף בנתיב הקריטי.

## 4.1 Session, Trace, Span

לפני שאפשר לנתח, צריך מודל נתונים. ריצת agent מתחלקת לשלוש רמות: Session היא השיחה כולה מתחילתה ועד סופה; Trace הוא מחזור אחד של שאלה ותשובה וכל מה שקרה באמצע; ו-Span הוא יחידת עבודה בודדת בתוך ה-Trace — קריאה ל-LLM, tool call, או חיפוש ב-Knowledge Base [17:03].




*ה-Trace מהתרחיש על ציר זמן, עם פירוק ה-spans ומטריקות הטוקנים לכל אחד מהם*

ה-Trace שהוצג על הבמה לקח 12 שניות ומורכב מ-span תכנון (קריאה ל-LLM), span קריאה ל-API של ה-CRM, span חיפוש ב-Knowledge Base, ולבסוף קריאת summarize. הממצא נמצא בשלישי: "אנחנו רואים פה בוורוד את החיפוש בנולדג' בייס, שאנחנו רואים גם שהוא לקח שלוש פעמים ולא חזר אף מסמך, וערך 7.4 שניות" [18:37]. כלומר גם מקור ההזיה וגם מקור ה-latency מופיעים באותה תמונה.

## AgentCore Observability 4.2

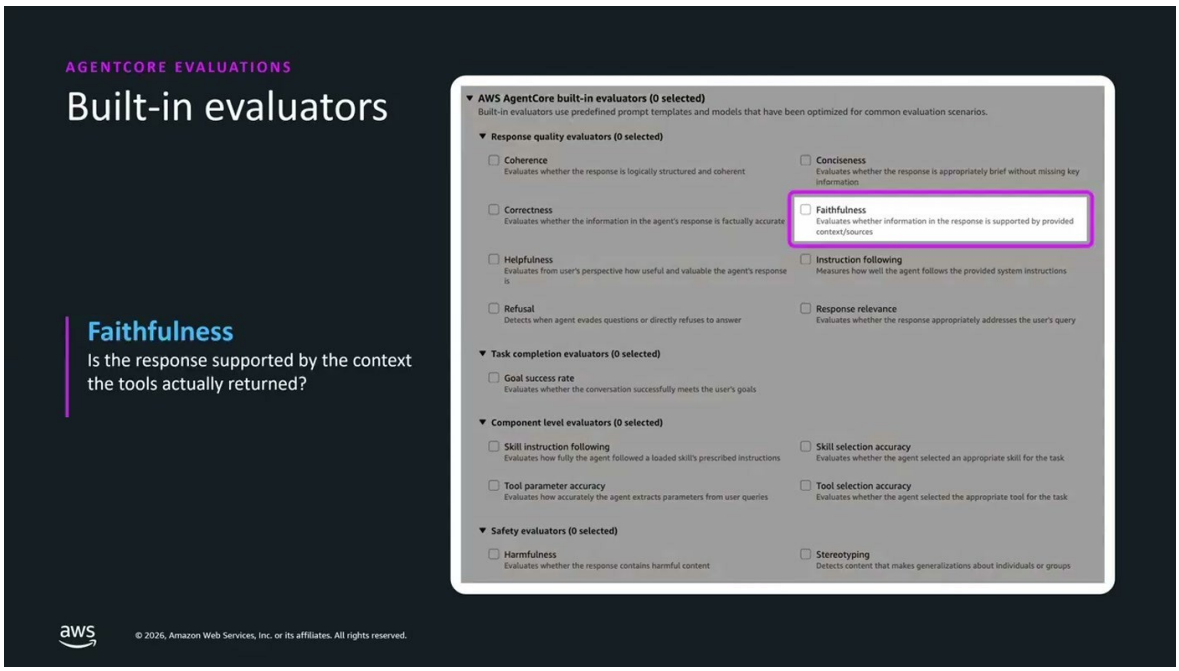
השירות שהוצג כמקום שבו כל זה נראה ב-AWS הוא AgentCore Observability: ויזואליזציה של התהליכים הפנימיים של ה-agent, דשבורדים מוכנים מראש, והטלמטריה זורמת בפורמט OpenTelemetry — "כך שאם יש לנו לקוחות שמשתמשים בכלים אחרים כמו למשל DataDog, LungFuse, Arise.ai או כלים נוספים שעובדים בפורמט הזה אפשר לעזרים את הטלמטריה הזאת בקלות גם לשם" [18:37].

בדמו הקצר שרץ ב-CloudWatch הוצגה ירידה מרמת כלל הסשנים של כל ה-agents, לסשן בודד, ל-Trace בודד ולטראג'קטוריה המלאה שלו, כולל Timeline View של ה-spans, כמות טוקנים in/out והזמן לכל span. הנקודה המעשית: "כל המידע הזה חי בקלאדווטש זה אומר שאני יכול להפעיל שאילטות של logs אינסייטס ואני יכול לייצר אלרטים על בסיס הדאטה הטלמטרי הזה" [18:37, 20:08].

# 5. Evaluation: האם מה שקרה היה נכון

המעבר בין שני החלקים נוסח כך: "אז אובזרבביליטי בעצם אומר לנו מה קרה? אבל אם מה שקרה זה בכלל דבר טוב ושאלה אחרת שצריך לשאול מהי ההגדרה לטוב" [20:08]. Evaluations הוגדרו כ"מדידה שיטתית של איכות התשובות", והנימוק לקיומן הוא קבלת החלטות: "כל שינוי מודל, כל החלפת פרומט, כל שינוי בכלים שהאג'נט משתמש בהם, אנחנו נרצה למדוד ולקבל החלטות מבוססות נתונים" [20:08].

שתי שאלות, שני כלים. בשלב שלפני הבנייה — "איזה מודל אנחנו בכלל מתחילים איתו?" — הכלי הוא Amazon Bedrock Evaluations, להשוואת מודלים על פרומפטים או על תהליכי RAG כנקודת פתיחה. אחרי שה-agent כבר רץ בפרודקשן השאלה משתנה: "האם הוא נותן ערך והאם הוא פותר את הבעיה שלשמה הוא נוצר" [20:08], וכאן נכנס AgentCore Evaluations, שמדרג את הטראג'קטוריה כולה על דאטה מפרודקשן ובזמן אמת, כשירות מנוהל שתוצאותיו זורמות ל-CloudWatch כמו כל מטריקה אחרת [21:41].




*מפת ה-evaluators המובנים, עם Faithfulness מודגש — האם התשובה נתמכת בקונטקסט שהכלים באמת החזירו*

Evaluation הוגדרה כאוסף של evaluators בשתי משפחות. ב-built-ins ההנחיות, ה-judging model וסולם הדירוג מוגדרים מראש, ולפי הספר זמינים 13 כאלה. הדוגמה שנבחרה היא Faithfulness, ובעברית שהוצעה על הבמה "נאמנות למקור": evaluator שבודק שהתשובה הסופית של ה-agent נשענת על הקונטקסט שחזר מה-tool call. בהחזרה לתרחיש מהפרק הקודם — אותו חיפוש שהחזיר אפס מסמכים — "היינו יכולים למנוע את התשובה הזויה הזאת שחזרה ללקוח" [23:13].

המשפחה השנייה, Custom Evaluators, מאפשרת להגדיר בשפה חופשית באנגלית גם את ההנחיות וגם את קריטריוני ההצלחה. הדוגמה שהוצגה מגדירה ל-agent שאסור לו לחשוף פרטי חוזה, הנחות, SLA-ים ונתונים רגישים אחרים ללא אסקלציה ואישור אנושי — ובמקביל מגדירה מהי אסקלציה מוצלחת, הגדרה שהודגש שהיא ספציפית לארגון ועשויה להשתנות מחברה לחברה [23:13].

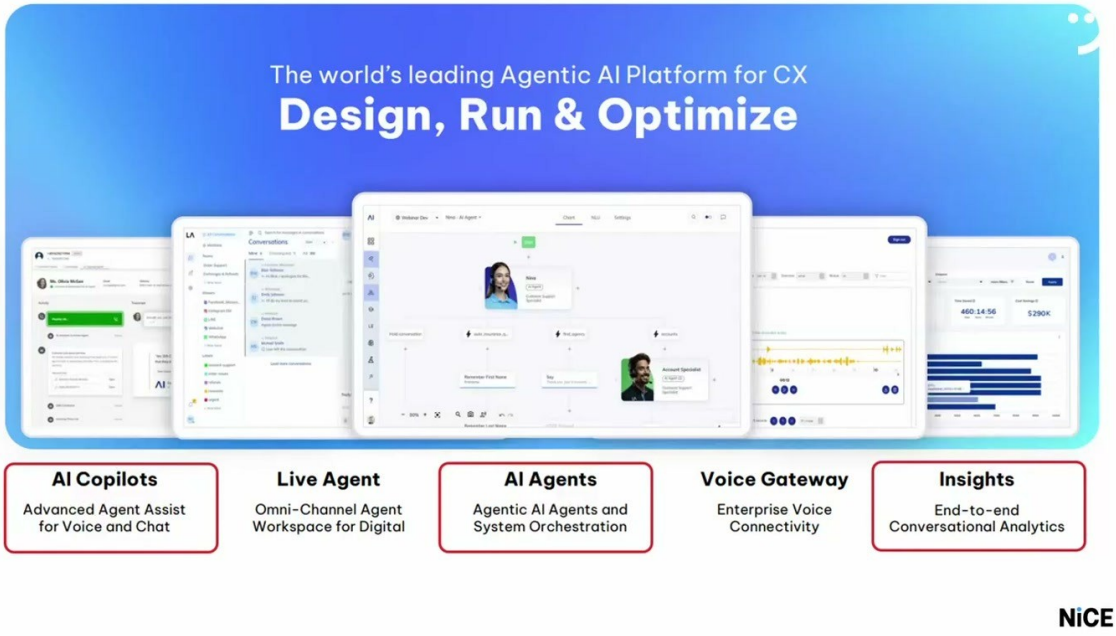
### שני מודים להרצה

- **Online Evaluation.** מגדירים sampling rate ודוגמים את התנועה החיה בפרודקשן באופן שוטף [23:13].
- **On-demand Evaluation.** אותה קונפיגורציה רצה בתוך תהליכי ה-CI/CD על דאטהסט מוכן, ויכולה למשל לחסום deployment שלא עובר threshold [23:13].

הסיכום של החלק הזה, ושל הסלייד שנשא את הכותרת Running ≠ Working: "אנחנו רואים שקיים איזשהו פער בין אייג'נט שהוא רץ לבין אייג'נט שהוא נותן ערך אמיתי שהוא מחזיר תשובות מדויקות ונכונות" [23:13].

# 6. המקרה של NICE: מהמטריקה להמלצה

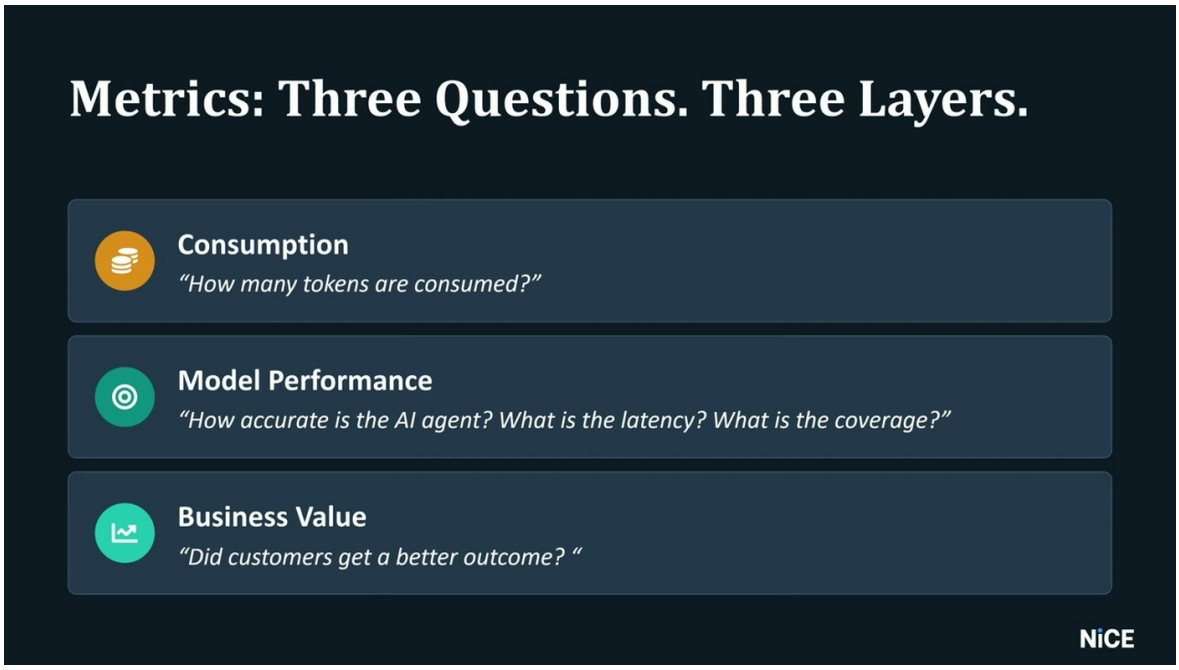
נעמה דמתי, VP Research ב-NICE, הציגה את המימוש בפועל. NICE היא חברה גלובלית המשרתת עשרות אלפי לקוחות בשלושה תחומים: Customer Experience, מניעת פשיעה פיננסית (NICE Optimize), ופתרונות ביטחוניים למגזר הציבורי (Public Safety). הפלטפורמה שהוצגה, Agentic AI Platform for CX, משרתת גם את הארגונים נותני השירות וגם את מיליוני הצרכנים שפונים אליהם.




*חמשת האזורים בפלטפורמה של NICE — AI Copilots, Live Agent, AI Agents, Voice Gateway ו-Insights*

שלושת האזורים שהודגמו: AI Co-Pilots — סוכנים שמסייעים לנציגי השירות האנושיים; AI Agents — בוטים שיחתיים שמטפלים בפניות בערוצי self-service; ו-Insights — פלטפורמת conversational analytics שמאפשרת לבעלי עניין עסקיים בארגון "להבין מה הסטטוס של אופרציית ה-AI, ולבצע שינויים והתאמות על מנת לקבל תוצאות יותר טובות" [26:20]. הדגש החוזר היה על שליטת הלקוח: בחירת מודל לכל use case, התאמות לחלקים מהפרומפטים, אימוץ המלצות אופטימיזציה, והגדרת פרמטרים ספציפיים לאבלואציה של ה-agents.

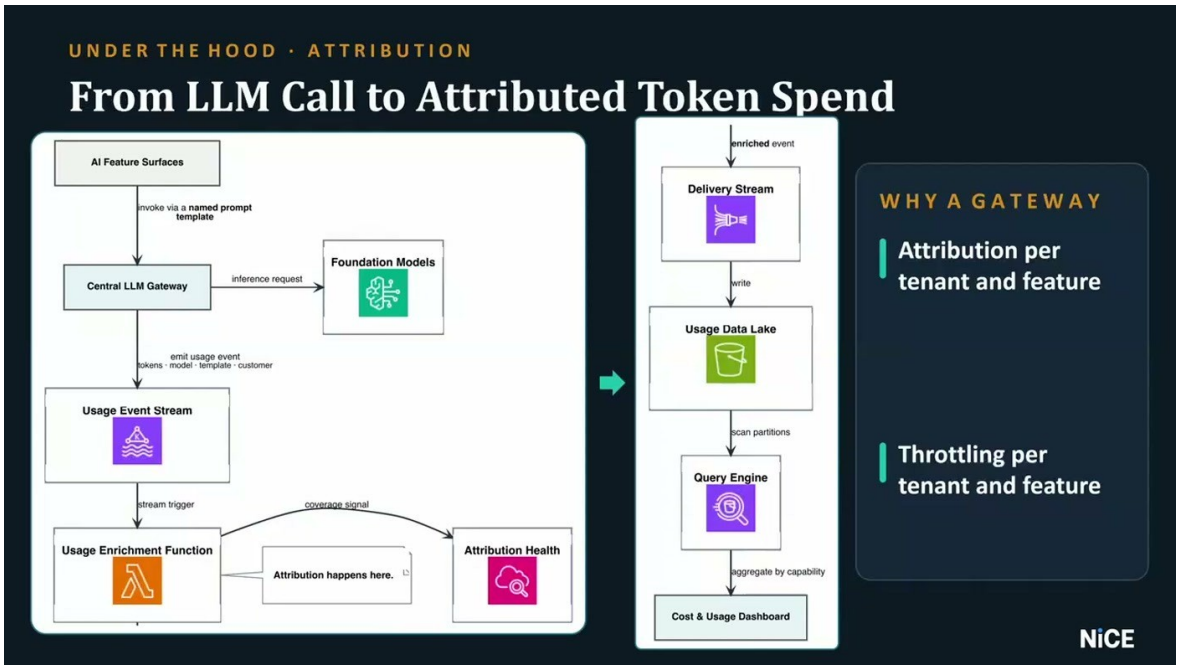
## 6.1 שלוש שכבות של מטריקות




*שלוש השאלות שכל שכבת מטריקות עונה עליהן — צריכה, ביצועי מודל, וערך עסקי*

- **Consumption.** כמה טוקנים נצרכו לכל use case, וכמה מהם input, output ו-caching [27:55].
- **Model Performance.** האיכות של ה-AI agents — accuracy, coverage ו-latency [27:55].
- **Business Value.** המטריקות שמאפשרות לבעלי העניין העסקיים להבין "האם למעשה קיבלתי את הוליו העסקי שציפיתי לקבל" מול מה שנצרך [27:55].

## 6.2 למה נבחר LLM Gateway



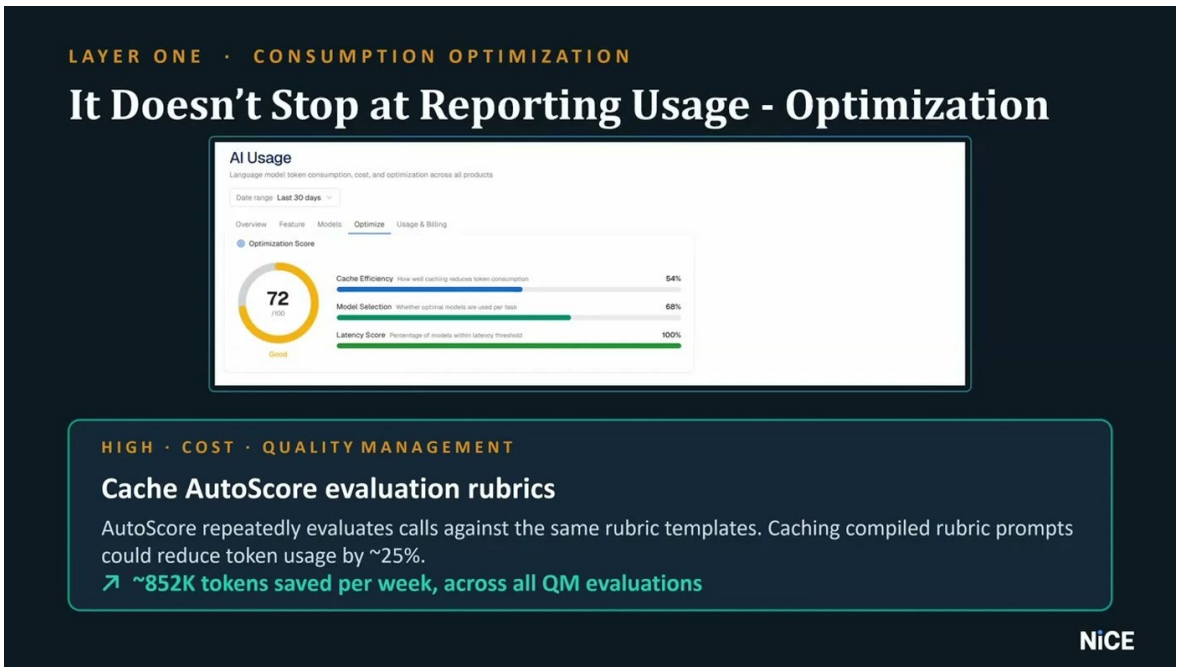

*הנתיב מקריאת LLM ועד ייחוס טוקנים — Gateway, usage event stream, העשרה, ודשבורד עלות ושימוש*

Usage Attribution הוגדר כ"היכולת לשייך כל טוקן וטוקן שהמערכת צרכה לאיזשהו "business capability [29:30]. מבין ארבע השיטות שארז הציג, NICE בחרה ב-LLM Gateway: מיקרו-שירות שכל בקשות ה-inference ל-Bedrock עוברות דרכו ומדווח את הנתונים לכל בקשה — כמות טוקנים, ה-prompt template, המודל, ה-latency, input ו-output — ל-usage stream שיודע למפות מ-prompt template ל-business capability.

הנימוק לבחירה נמסר בשתי דרישות. הראשונה היא גרנולריות: "רצינו ממש להיות מסוגלים לשייך כל בקשה גם לטננט וגם לביזנס קייפביליטי, לפיצ'ר" — ו-Application Inference Profile לא איפשר זאת בלי להגדיר יותר מדי פרופילים [29:30]. השנייה היא שליטה: יכולת לבצע throttling ברמת ה-tenant והפיצ'ר, בעוד ש"היום [throttling] בבדרוק מתאפשר ברמת המודל והאקאונט, ואנחנו רצינו גונולריטי אחר" [31:04].

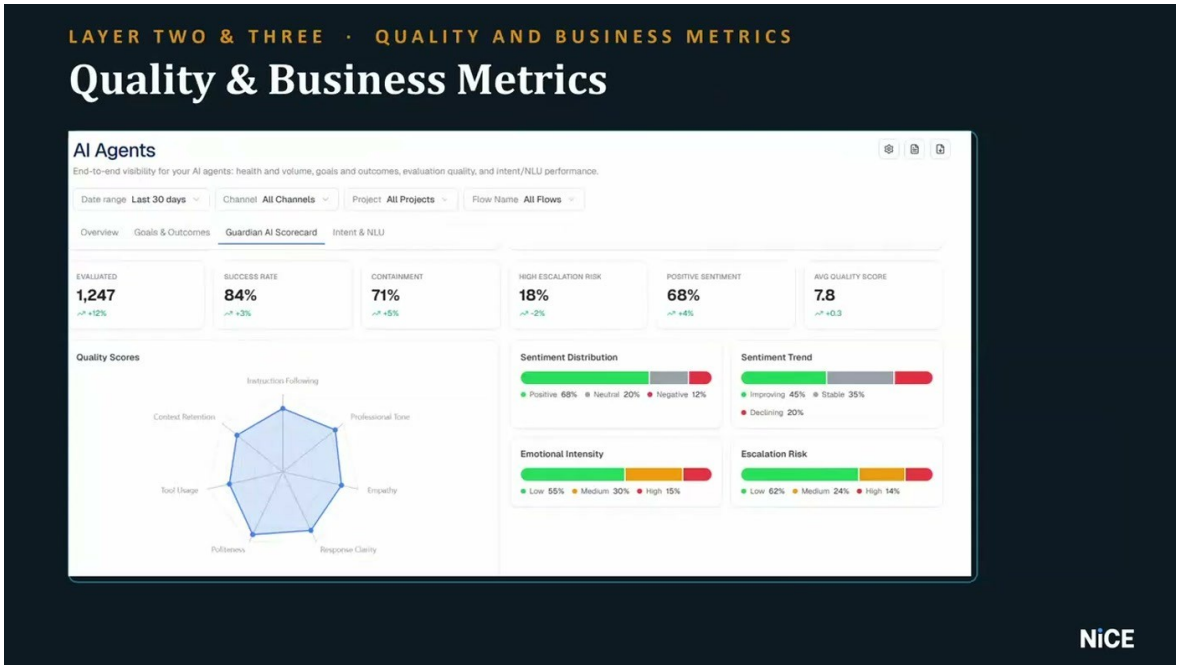
## 6.3 מדשבורד להמלצה

הנקודה שבה הסיפור נהיה מעניין היא לא איסוף המטריקות אלא מה שנעשה איתן: "אנחנו לא עוצרים רק להראות מטריקות בדשבורד, אנחנו גם מנתחים את המידע הזה ומביאים המלצות לאופטימיזציה" [31:04].



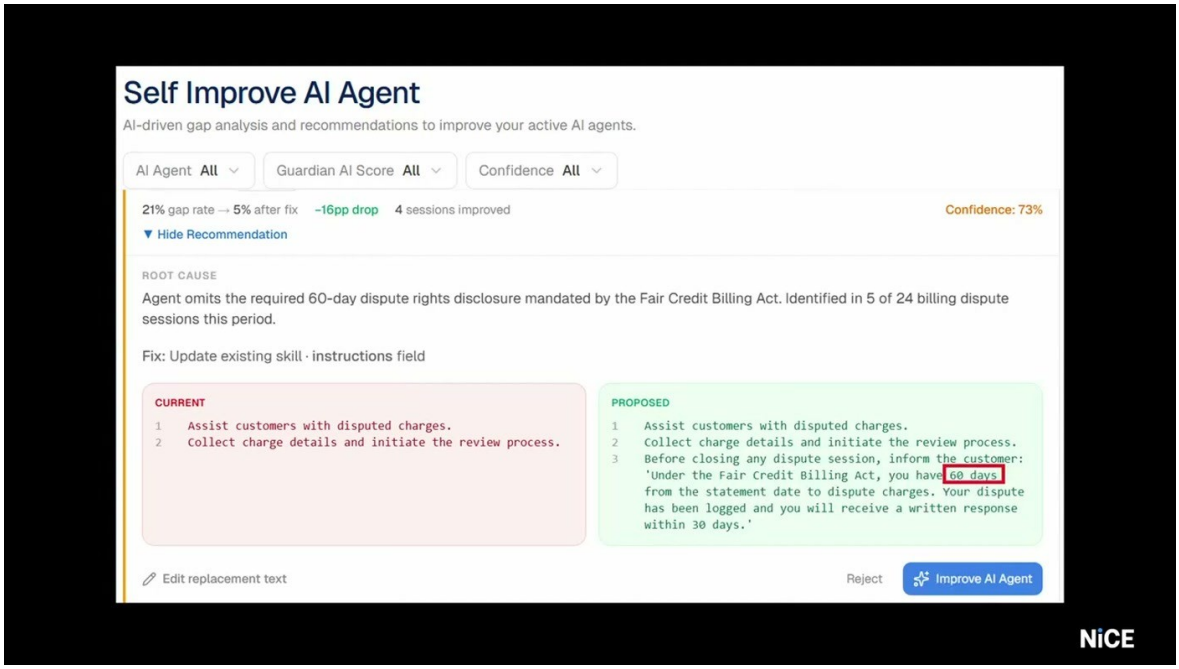

*המלצת caching שנגזרה מניתוח השימוש בפיצ'ר AutoScore, עם ההערכה המספרית שעל הסלייד*

הדוגמה שהוצגה חוזרת ישירות לפרק האופטימיזציה של מור: ניתוח השימוש בפיצ'ר AutoScore אצל tenant מסוים הוביל למסקנה ש-caching יקטין משמעותית את צריכת הטוקנים [31:04]. על הסלייד עצמו מופיעה ההערכה: הפחתה של כ-25% בצריכת הטוקנים, וכ-852K טוקנים שנחסכים בשבוע על פני כלל הערכות ה-QM.



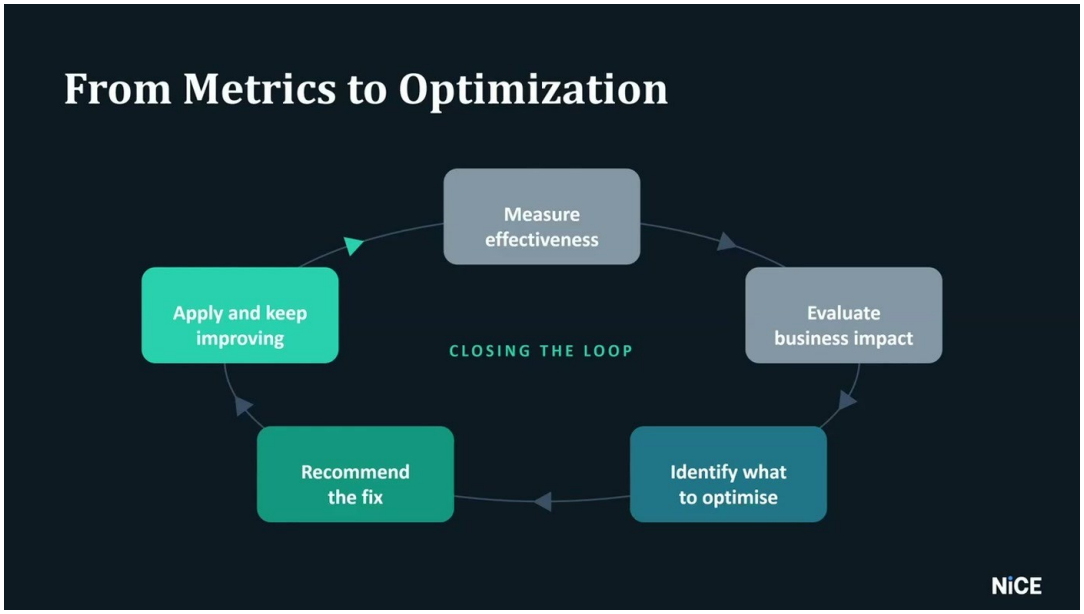

*הדשבורד שמאחד מטריקות עסקיות ומטריקות איכות — ובצד שמאל ה-spider chart של פרמטרי ההערכה*

שתי השכבות האחרונות אוחדו במכוון לדשבורד אחד. בדוגמה שהוצגה: 1,247 סשנים של לקוחות קצה שהגיעו ל-AI agents, ולצדם מטריקות עסקיות. על מדד ה-containment נאמר: "זה אומר ש71 אחוז מאותם סשנים סיימו אצל אותו conversational bot, אבל 29% עברו אסקלציה ל-human agent, כמובן שזה יכול להוות איזושהי בעיה" [32:38]. לצדן, מטריקות איכות — instruction following, טון ופרמטרים נוספים — "את זה אנחנו מיישמים באמצעות LLM as a judge", עם אפשרות ללקוח להוסיף קריטריון משלו. הדוגמה שניתנה: לקוח שהוסיף קריטריון Brand Alignment כדי לוודא שה-agent משתמש בטרמינולוגיה הנכונה של המותג [32:38].




*ההמלצה כפי שהיא מוצגת לאישור — ה-instructions הקיימים מול המוצעים, זה לצד זה*

ואז חוזר התרחיש של ארז — ה-agent שהבטיח החזר בסתירה למדיניות. הניתוח ב-NICE זיהה ששני AI agents סובלים מאותה סתירה בין מה שהם מבטיחים לבין המדיניות הארגונית, והמערכת ייצרה המלצה לשינוי ספציפי ב-instructions שלהם. בעלי העניין העסקיים רואים משני צדי המסך את ההנחיות הקיימות מול המוצעות, יכולים להריץ סימולציה על ההמלצה, ולבסוף לאשר ולפרסם גרסה חדשה של ה-agent [34:08].




*הלולאה כפי ש-NICE מציגה אותה — מדידה, זיהוי מה לשפר, המלצה, יישום, וחזרה למדידה*

"ראינו ממש איך אנחנו סוגרים 360 מעלות את כל הנושא הזה אנחנו הוספים הרבה מאוד מטריקות אנחנו חוץ מלהנגיש אותם ללקוחות אנחנו מנתחים אותם אנחנו ממליצים לעשות אופטימיזציות כאלה ואחרות" [34:08]. מור

סיכמה את החלק הזה בכך ש-NICE "לא רק הסתפקתם באיסוף של הדאטה והצגתו בגרפים ודשבורדים מעניינים, אלא תרגמתם אותו להמלצות אופרטיביות שהלקוחות שלכם יכולים לממש בלחיצת כפתור" [35:41].

# 7. מה לוקחים מכאן

המסר המרכזי של הסיכום היה שזו אינה יוזמה חד-פעמית: "לא מדובר בפרויקט חד פעמי, מדובר בבנייה של תשתית, מחזורית, שכל הזמן חוזרת על עצמה. אנחנו עושים Observability, מנטרים את הפעילות, מעטרים את המקומות שבהם אנחנו יכולים לעשות אופטימיזציה, בוחנים שינויים, מטמיעים, ממשיכים לנטר וחוזר חלילה" [35:41].

- **התחילו במה שכבר בייצור.** Prompt Caching ו-Inference Tiers לא דורשים שינוי במודל או בארכיטקטורה, והמספרים שהוצגו להם הם הגדולים ביותר בסשן ביחס למאמץ הנדרש.
- **בחרו את יחידת הפילוח לפני שבוחרים כלי.** ארבע שיטות ייחוס העלות נבדלות ברזולוציה ובמחיר התפעולי. NICE ירדה ל-Gateway רק משום שנדרשה רזולוציה של tenant ופיצ'ר, ויכולת throttling באותה רמה.
- **ודאו שיש לכם Trace ולא רק לוג.** שורת הלוג בסלייד של ארז היא מבחן מעשי: אם אחרי תלונת לקוח אי אפשר לומר אילו מסמכים נשלפו ואילו כלים הופעלו, אין תשתית Observability אג'נטית.
- **Evaluation צריך לרוץ בשני מקומות.** On-demand בתוך ה-CI/CD כשער לפני deployment, ו-Online בדגימה שוטפת של תנועת פרודקשן — כי, כפי שנאמר בסיכום, צריך לוודא שהביצועים נשמרים "גם כשהעולם בחוץ משתנה" [37:12].
- **שלבו מטריקה עסקית בתוך תמונת ה-Observability.** ההמלצה החוזרת בסשן היא לא להסתפק בטוקנים ו-latency. מדד ה-containment של NICE הוא הדוגמה: מספר שאומר מיד האם שילמנו פעמיים על אותה פנייה.
- **היעד הוא המלצה, לא דשבורד.** ההבדל בין המקרה של NICE לבין רוב מימושי ה-Observability הוא שהדאטה מתורגם להמלצה קונקרטית עם סימולציה ואישור אנושי לפני פרסום גרסה.

### נקודות שלא נענו בסשן

| נושא | מה חסר | חותמת |
|---|---|---|
| עלות ה-Evaluation עצמה | LLM as a judge על תנועה חיה צורך טוקנים; שיעור הדגימה והעלות לא כומתו | 23:13 |
| בחירת ה-judging model | הוזכר כחלק מהגדרת ה-evaluator, בלי המלצה מעשית | 21:41 |
| אבטחת ה-Gateway | הוזכרה כסיכון עם הפניה לאירוע בתעשייה, בלי דפוס מיטיגציה | 17:03 |
| מספרי הביצועים | 50%/85%/90% הוצגו כערכי שיא של השירות; לא הוצגה מדידה מהשטח | 9:15 |

# 8. מונחון

| מונח | משמעות |
|---|---|
| Prompt Caching | סימון החלק הקבוע בבקשה כך שיישמר במטמון ולא יישלח מחדש בכל קריאה |
| Inference Tiers | שכבות הרצה ב-Bedrock — Standard, Priority, Flex, Batch — לאותו מודל במחירים וזמני תגובה שונים |
| Open Weights | מודלים שהמשקולות שלהם זמינות לשימוש ול-fine-tune, אך לא בהכרח הקוד, הדאטה או רישיון מסחרי חופשי |
| Session / Trace / Span | שלוש רמות של ריצת agent: השיחה כולה, מחזור שאלה-תשובה בודד, ויחידת עבודה אחת בתוכו |
| AgentCore Observability | שירות AWS לווִיזואליזציה של התהליכים הפנימיים של agent, עם ייצוא ב-OpenTelemetry |
| AgentCore Evaluations | שירות מנוהל לדירוג טראג'קטוריות של agent על דאטה מפרודקשן; תוצאותיו זורמות ל-CloudWatch |
| Application Inference Profile | פרופיל מתויג שמצביע על מודל, מועבר ל-agent כ-ARN, ומייצר שורת חיוב נפרדת |
| LLM Gateway | רכיב ביניים שכל קריאות ה-LLM עוברות דרכו, לצורך ייחוס, מכסות ו-throttling |
| Faithfulness | evaluator שבודק אם התשובה נתמכת בקונטקסט שהכלים באמת החזירו |

| מונח | משמעות |
| --- | --- |
| LLM as a judge | שימוש במודל שפה כמדרג איכות של פלטי מודל אחר, לפי הנחיות וסולם דירוג |
| Containment | שיעור הסשנים שהסתיימו אצל הבוט בלי אסקלציה לנציג אנושי |
| Usage Attribution | שיוך כל טוקן שנצרך ליחידה עסקית — tenant, פיצ'ר או capability |