מי בזבז מה? שיוך עלויות GenAI על AWS

TLV-DEM310 — סיכום סשן, AWS Summit Tel Aviv 2026

Eitan Sela, Principal GenAI/ML Specialist Solutions Architect, AWS

10 בספטמבר 2026 | משך: כ-26 דקות | הופק מהקלטת הסשן

מסלול: AWS Demos | רמה: 300 — Advanced

על המסמך הזה

המסמך מסכם את הסשן TLV-DEM310 מתוך AWS Summit Tel Aviv 2026, בהתבסס על ההקלטה המלאה שפורסמה באתר הסאמיט. זהו סשן דמו קצר וצפוף: כמעט כל דקה בו היא מסך קונסול או קטע קוד אמיתי. המסמך נבנה מתמלול אוטומטי של ההקלטה ומהסליידים ומסכי הדמו שחולצו מהווידאו עצמו, כך שאפשר לחזור לתוכן — ובעיקר לרשימת השיטות ולהבדלים ביניהן — בלי לצפות שוב בהקלטה. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה הוצג ומה המסקנה המעשית.
- **פרקים 2–4 — הרקע והמפה:** למה שיוך עלויות GenAI קשה, מה זה Mantle, ושתי השאלות שקובעות איזו שיטה מתאימה.
- **פרקים 5–11 — שבע השיטות:** כל שיטה בפרק משלה, עם מסך הדמו שהוצג ועם המחיר שלה בזמן ובשינויי קוד.
- **פרקים 12–13 — ההשוואה ומה הלאה:** טבלת ההשוואה שסגרה את הסשן, וארבעת הצעדים להתחלה.
- **פרק 14 — מילון מונחים:** המונחים שהוצגו בסשן, באנגלית ובעברית.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך מונחים לועזיים הופיעו בו בשיבוש פונטי ותוקנו ידנית מול הסליידים ומסכי הדמו ("מנטל" → Mantle, "בדרוק" → Bedrock, "ליטל אליהם" → LiteLLM). הציטוטים אומתו מול ההקלטה בחותמת הזמן המצוינת. מספרים, שמות משאבים וערכי תגים שמופיעים במסמך נלקחו ממסכי הדמו עצמם.

1. תקציר מנהלים

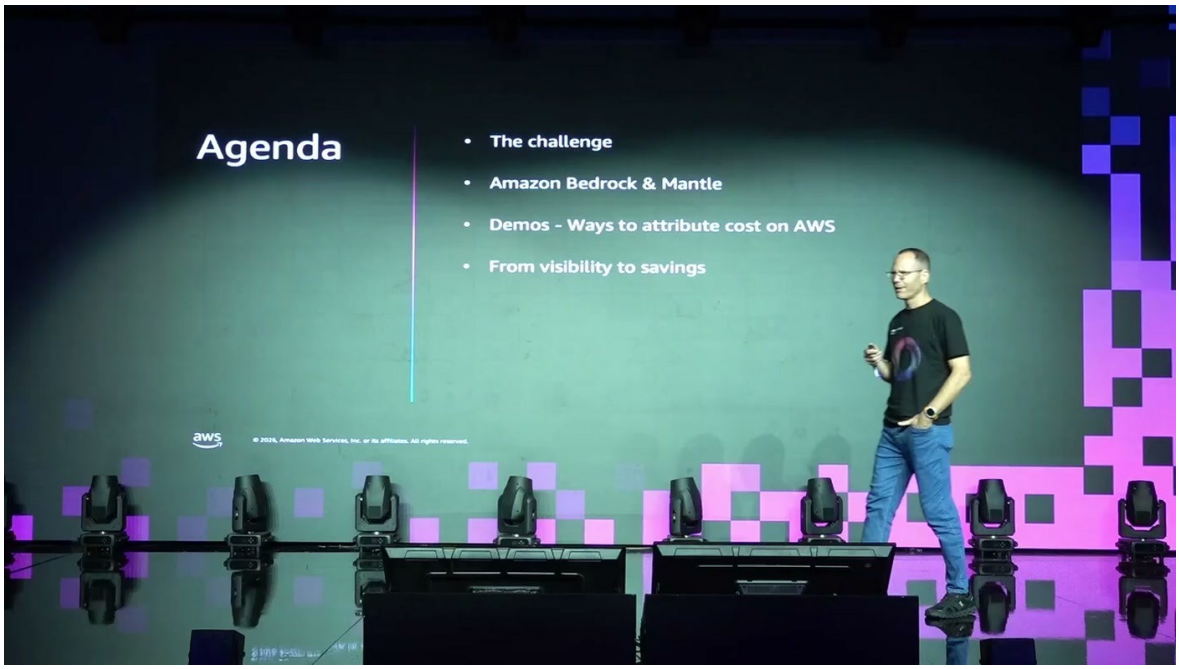
Eitan Sela, Principal GenAI/ML Specialist Solutions Architect בצוות הישראלי של AWS, פתח בבעיה שהוא שומע בכל מפגש לקוחות: חשבונית AWS שבה סעיף Bedrock גדול שאי אפשר להסביר מאיפה הגיע. הסשן כולו הוא תשובה מכניסטית אחת לשאלה הזאת — שבע דרכים קונקרטיות לשייך כל דולר של inference לישות שצרכה אותו, עם דמו חי לכל אחת מהן.

- **הבעיה היא שהמודל הוא משאב משותף.** בניגוד ל-workload קלאסי שנמדד בשעות CPU או בטרה-בייטים, כאן כולם — מפתחים, אפליקציות ו-agents — קוראים לאותם מודלים דרך אותו endpoint. "כל אפליקציה או מפתחים קוראים למודלים, לאותם מודלים בעצם, אז זה shared כזה resource וקשה לשייך" [3:01].
- **שבע שיטות, מהן שש native ואחת proxy.** IAM Principal, Application Inference Profiles, Workspaces, Projects, Per-Request Metadata, IAM Identity Logs — ומעליהן AI Gateway מסוג LiteLLM. השיטות אינן חלופיות אלא מכסות שאלות שונות.
- **שתי שאלות קובעות איזו שיטה מתאימה.** ראשית, איזה Bedrock runtime endpoint — הוותיק או Mantle החדש. שנית, לפי מה משייכים — WHO (זהות/מפתח), WHAT (אפליקציה או agent), או WHICH CALL (קריאה בודדת).
- **הפער האמיתי הוא זמן, לא דיוק.** ארבע מהשיטות מייצרות ערך דולרי אמיתי בבילינג, אבל בעיכוב של עד 24 שעות — "וערך דולרי ב-AWS יש דילי של עד 24 שעות" [6:03]. שתיים מייצרות ספירת טוקנים דרך Model Invocation Log בעיכוב של כדקה, ומאפשרות חסימה בזמן אמת — במחיר של חישוב עצמאי של טוקנים כפול מחיר מודל.
- **לשלוש מהשיטות יש מחיר בקוד.** IAM Principal Attribution לא דורש שינוי קוד כלל; Application Inference Profiles דורש להחליף את ה-modelId ב-ARN של הפרופיל; Workspaces/Projects דורש פרמטר אחד ב-client; Per-Request Metadata דורש להוסיף אובייקט JSON לכל קריאה.
- **השיוך הוא רק השלב הראשון.** הסשן הסתגר בכוונה על attribution ולא על optimization: "Attribution first. Optimization next" — כי לדברי המרצה, לזהות את הצרכן זה האתגר, והטיפול (בחירת מודל, prompt caching, cross-region inference) הוא כבר עולם ידוע.

רלוונטיות לארגון גדול

שני התרחישים שהמרצה חזר אליהם הם בדיוק התרחישים של ארגון ריכוזי: מפתחים שמריצים Claude Code או Codex מול Bedrock וצריך לתת להם תקציב אישי ולחסום אותם כשהוא נגמר, ו-agents בפרודקשן שצריך לחייב עליהם מרכז עלות. לשני המקרים יש בסשן תשובה שונה — הראשון מחייב מסלול near-real-time (Identity Logs או Gateway), השני מסתדר יפה עם תגים בבילינג. הפער שהמרצה הודה בו במפורש: חסימה בזמן אמת מבוססת טוקנים דורשת חישוב עלות עצמאי, ואין UI מובנה של תקציבים פר-מפתח — שם נכנסים מוצרי הצד השלישי.

2. האתגר — למה שיוך עלויות GenAI שונה



סדר היום בארבעה שלבים: האתגר, Bedrock ו-Mantle, הדמואים, ומעבר מוויזיביליות לחיסכון

הפתיחה הייתה תרחיש ולא הגדרה: "אתם משתמשים ב-Claude Code, ב-Codex, יש לכם agents בפרודקשן שרצים, ואתם מסתכלים על החשבונית וקשה להבין מי בזבז מה" [0:00]. המרצה הציג את עצמו כמי שמגיע מרקע של יותר משני עשורים בפיתוח תוכנה ו-Machine Learning, ותפקידו לעזור ללקוחות AWS לבנות מערכות GenAI.

ההסבר לקושי הוא מבני. workload מסורתי ב-AWS נמדד ביחידות שמצמידות עצמן למשאב: שעות CPU של EC2, טרה-בייטים של S3. ב-GenAI היחידות אחרות ונזילות יותר — "פה יש token in, token out, יש cache token, יש כל מיני סוגי מודלים" [3:01] — והצרכנים כולם חולקים את אותו משאב לוגי.

> **— [3:01]** וזה עוד יותר מסתבך, כי יש לי מפתחים שמשתמשים ב-Claude Code, ויש לי agents בפרודקשן שרצים, וכל מיני אפליקציות GenAI ו-RAG, וקשה מאוד לשייך את העלויות

המרצה סימן את התחום כצעיר: "וזה באמת אתגר שהוא מאוד גדול ואנחנו רק בחיתולים שלו" [3:01]. זהו הרקע שמסביר למה יש שבע שיטות ולא אחת — אף אחת מהן איננה פתרון כולל.

3. Amazon Bedrock ו-Mantle

לפני הדמואים הוצג הבסיס. Bedrock הוצג כפלטפורמת ה-AI של AWS — מאות מודלים ב-serverless, תשלום לפי token in / token out, ללא ניהול תשתית. המרצה ציין שכל המודלים רצים על תשתית EC2 של AWS, שהדאטה הנשלח אליהם אינו נשמר ואינו נשלח לספק אחר, ושאפשר לצמצם ל-region ספציפי ולהשתמש ב-VPC Endpoint כדי שהתעבורה לא תצא מרשת AWS [4:32]. הוא הזכיר גם שמודל OpenAI חדש הושק יומיים לפני הסשן ונוסף לקטלוג; במסכי הדמו לאורך הסשן מופיע מזהה המודל global.openai.gpt-5.6.

מה זה Mantle

כשנשאל הקהל מי מכיר את Mantle, כמעט אף אחד לא הרים יד — ולכן הוקדש לו הסבר נפרד. Mantle הוא inference engine חדש שנבנה ל-Bedrock, שנועד לאפשר עבודה מאובטחת יותר ובסקייל עם אותם מודלים. שני ההבדלים המעשיים שהוצגו: תאימות SDK, ומושגי ניהול שאולים מעולם ספקי ה-LLM.

> **— [4:32]** אם יש לכם workload כבר שעובד עם OpenAI או Anthropic, אז פשוט לוקחים את הקליינט, משנים את ה-Base URL, וזהו, אפשר לעבוד עם זה

ההבדל השני הוא שב-Mantle קיימים המושגים Project (מ-OpenAI) ו-Workspace (מ-Anthropic), ושניהם ניתנים לתיוג. ההשלכה לשיוך עלויות היא ישירה: שתיים מתוך שבע השיטות קיימות רק ב-Mantle, ושתיים אחרות

רק ב-runtime הקלאסי. בקונסול הדבר גלוי — ה-navigation של Bedrock מסומן ב-"bedrock-runtime endpoint" או ב-"bedrock-mantle endpoint" בהתאם.

4. שתי השאלות ושבע השיטות

המרצה הציג מסגרת החלטה בת שתי שאלות. השאלה הראשונה טכנית: מול איזה endpoint האפליקציה עובדת — runtime או Mantle. השאלה השנייה עסקית: לפי מה רוצים לשייך.

- **WHO — זהות.** מי האדם או ה-principal שצרך. הרלוונטי למפתחים שמריצים כלי קידוד מול Bedrock.
- **WHAT — אפליקציה או agent.** איזה שירות GenAI, microservice או agent אחראי לעלות.
- **WHICH CALL — קריאה בודדת.** הרזולוציה הגבוהה ביותר: איזה tenant, פיצ'ר או session יצר את הקריאה.

הרזולוציה השלישית הוצגה עם נימוק עסקי מפורש: חברות SaaS עם אלפי לקוחות שאינן יודעות איך הלקוחות שלהן משתמשות ביכולות ה-GenAI, ולכן מגלמות זאת כתוספת אחוזים ברישיון.

> **[6:03]** ואז הם אומרים לי איתן, אנחנו מחייבים אותם עוד עשר עשרים אחוז מעלות כזה של הרישיון — אבל אפשר ממש פר קריאה להבין איזה tenant קרא למודל ולחייב אותו ממש בהתאם

שבע המכניקות כפי שהוצגו: ארבע צד שמאל בטור אחד, שלוש בשני — כולל LiteLLM Gateway כשיטה החיצונית היחידה

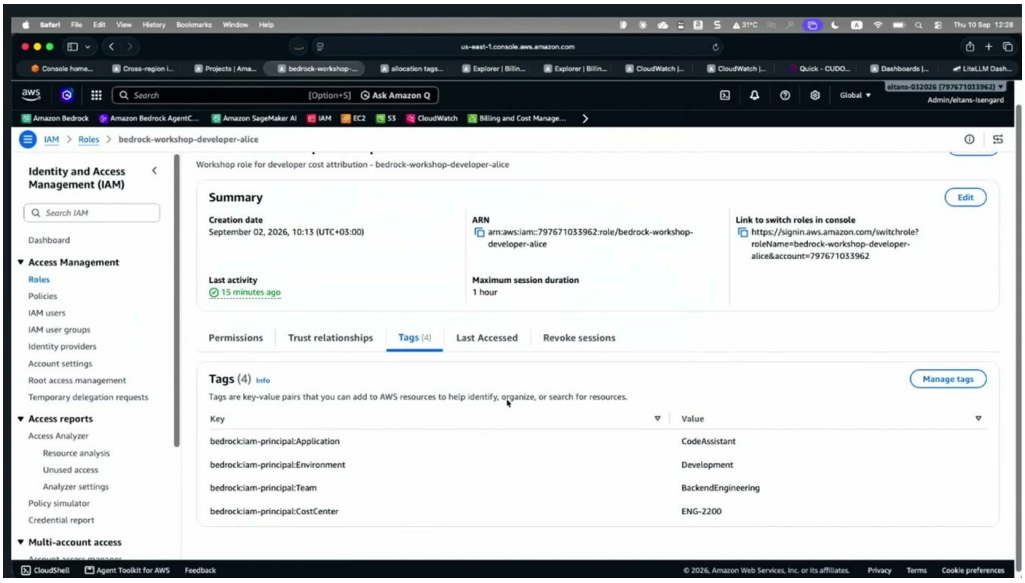
ההבחנה הקריטית שהוצגה כבר כאן, ושחזרה בכל פרק: חלק מהשיטות מייצרות ערך דולרי בבילינג, וכאלה סובלות מעיכוב של עד 24 שעות; חלק מייצרות ספירת טוקנים דרך Bedrock Model Invocation Log, ואלה near-real-time בעיכוב של כדקה.

כל הדמואים מבוססים על סדנה רשמית שנבנתה בישראל: "עשינו אותו בארץ, הוא כבר רץ בעולם, בארצות הברית, אירופה, APJ, שזה יפן ואוסטרליה" [7:41]. הסדנה — Track and Optimize Generative AI Spend on AWS — זמינה בקטלוג הציבורי של AWS Workshops ומכילה מודול נפרד לכל אחת מהשיטות, עם ריפוזיטורי קוד נלווה.

5. שיטה 1 — IAM Principal Attribution

המטרה: WHO. מעקב אחרי מפתחים או זהויות. הפלט: ערך דולרי בבילינג. העיכוב: עד 24 שעות. שינויי קוד: אפס.

המנגנון מבוסס על תגים שמוצמדים ל-IAM role. בדמו הוגדרו שלושה roles — bedrock-workshop-developer-alice, bob ו-carol — ולכל אחד ארבעה תגים תחת מרחב השמות bedrock:iam-principal. בקונסול נראה בבירור ש-Alice נושאת Application=CodeAssistant, Environment=Development, Team=BackendEngineering ו-CostCenter=ENG-2200.

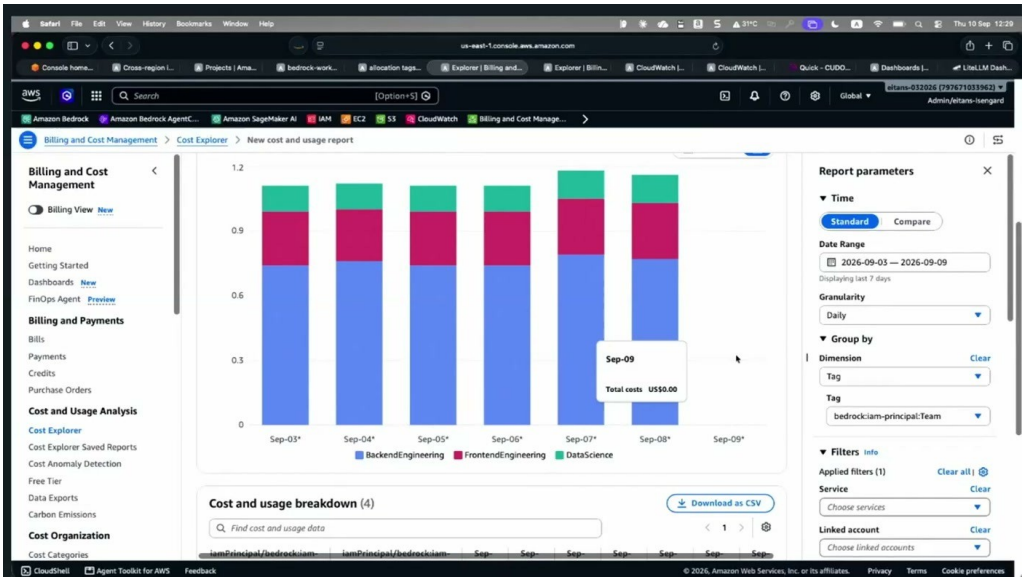


*ארבעת התגים על ה-role של Alice בקונסול IAM, תחת המפתחות bedrock:iam-principal:**

הקריאה עצמה נשארת רגילה לחלוטין. המרצה הדגיש: "פה אני לא עושה שינוי קוד, כי אם זה אפליקציה או מפתח אז משתמשים ב-Claude Code עם role... וקראתי ל-Bedrock באופן רגיל לגמרי עם Converse API" [9:13]. ה-assume role בדמו נעשה רק כדי לדמות את המפתחים.

> **מלכודת תפעולית** לתגים חדשים ב-AWS יש שני מחסומי זמן, לא אחד: ראשית הם צריכים לחלחל, ושנית צריך להפעיל אותם ידנית כ-cost allocation tags — הם מופיעים תחילה כ-Inactive. "צריך להבין שברגע שמשתמשים בתגים חדשים ב-AWS, צריך לחכות עד 24 שעות שהם יחלחלו" [9:13].

התוצאה הסופית היא דוח Cost Explorer מקובץ לפי התג bedrock:iam-principal:Team, שמראה פילוח יומי בין BackendEngineering, FrontendEngineering ו-DataScience. המרצה ציין שהוא בחר לסכם ברמת הצוות ולא ברמת המפתח הבודד.



Cost Explorer מקובץ לפי תג הצוות — שלושה צוותים, פילוח יומי, ערכי דולר אמיתיים

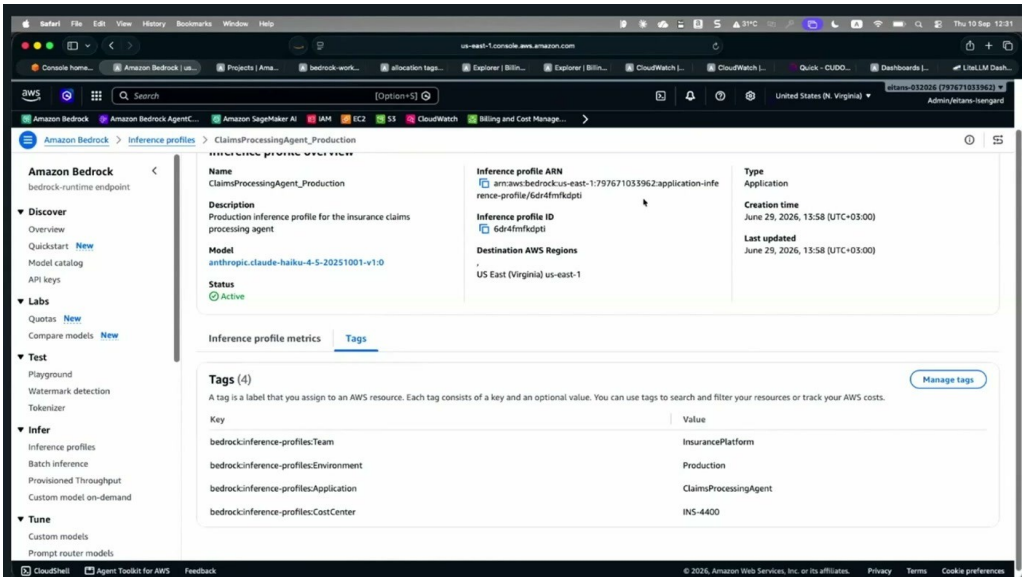
המגבלה נאמרה במפורש ומיד: "עכשיו חשוב להגיד — זה עד 24 שעות, ככה זה פחות מתאים למקרים שאנחנו רוצים לעקוב אחרי מפתחים בזמן אמת ולחסום אותם אם הם עברו תקציב מסוים" [9:13]. התשובה למגבלה הזאת היא שיטה 6, שתוצג בהמשך.

6. שיטה 2 — Application Inference Profiles

המטרה: WHAT. מעקב אחרי אפליקציות, agents ו-microservices. הפלט: ערך דולרי. העיכוב: עד 24 שעות. שינויי קוד: כן — החלפת ה-modelId.

ב-Bedrock קיימים inference profiles מוגדרי-מערכת — ה-modelId הרגיל שמעבירים בקריאה, למשל global.anthropic.claude-opus-4-5. מעליהם אפשר ליצור application inference profile משלך, למשל אחד לכל agent. בדמו הוגדרו שלושה: ClaimsProcessingAgent, PolicyRecommendation ו-FraudDetectionAgent.

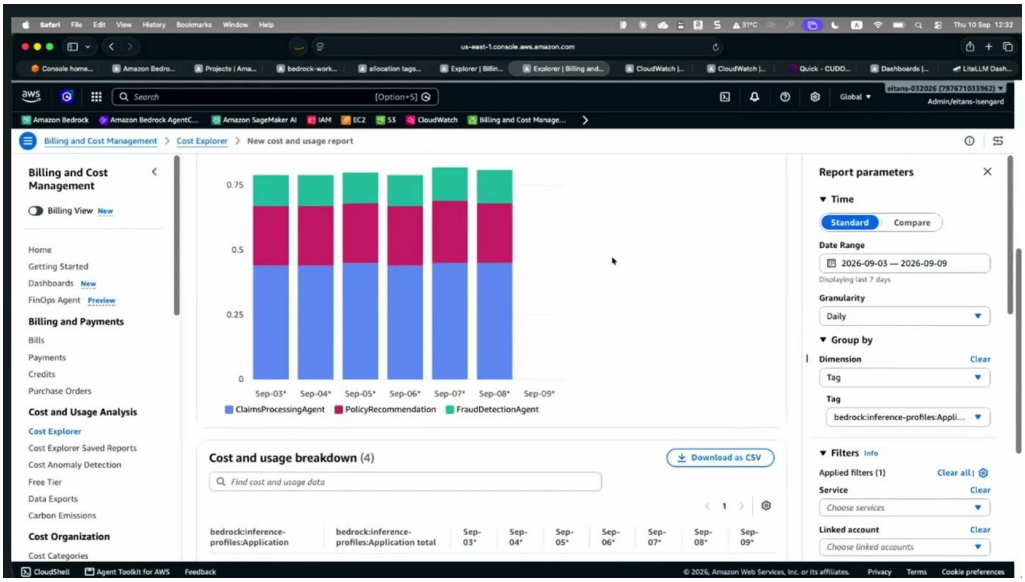
כל פרופיל כזה משויך למודל ספציפי — בדוגמה anthropic.claude-haiku-4-5 — ונושא ארבעה תגים משלו תחת bedrock:inference-profiles. הפרופיל שהוצג, ClaimsProcessingAgent_Production, נושא Team=InsurancePlatform, Environment=Production, Application=ClaimsProcessingAgent ו-CostCenter=INS-4400.



פרופיל ה-inference של סוכן עיבוד התביעות: ה-ARN, המודל המשויך, וארבעת התגים בלשונית Tags

כאן, בניגוד לשיטה הקודמת, יש מחיר בקוד — אבל מינימלי: במקום מזהה מודל מעבירים את ה-ARN של הפרופיל באותו פרמטר modelId. המרצה ניסח זאת ישירות: "אני משתמש ב-modelId שזה ה-inference profile ARN — ה-ARN הזה זה אומר שכן אני צריך לעשות שינוי קוד" [10:44]. שאר הקריאה, כולל Converse API, נשארת זהה.

מכאן המסלול זהה לשיטה הראשונה: התגים מחלחלים, מופעלים כ-cost allocation tags, ומופיעים ב-Cost Explorer — הפעם בפילוח פר-agent.

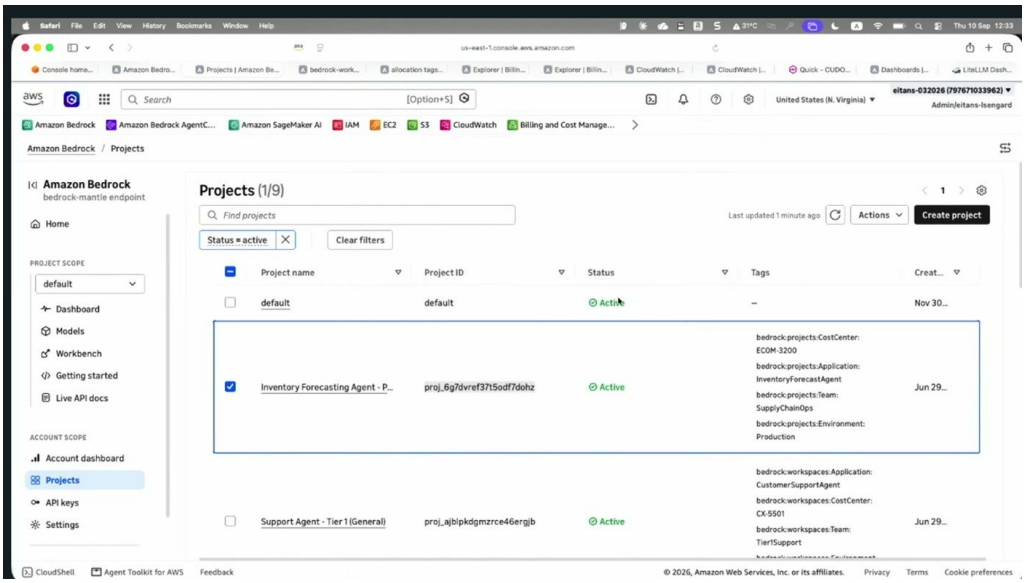


אותה תצוגת Cost Explorer, הפעם מקובצת לפי תג ה-Application של פרופילי ה-inference — עלות יומית לכל סוכן בנפרד

7. שיטות 3–4 — Workspaces ו-Projects ב-Mantle

המטרה: WHAT, אבל ב-endpoint אחר. הפלט: ערך דולרי. העיכוב: עד 24 שעות. שינויי קוד: פרמטר אחד ב-client.

מכיוון ש-Mantle תואם ל-SDK של OpenAI ושל Anthropic, הוא ירש את מושגי הניהול שלהם: Project ב-OpenAI, Workspace ב-Anthropic. שניהם נחשפים כאובייקטים בקונסול, עם מזהה ועם תגים — ולכן שניהם משמשים כמנגנון שיוך. בקונסול נראה פרויקט בשם Inventory Forecasting Agent עם Project ID proj_6g7dvref37t5odf7dohz ותגים תחת bedrock:projects — CostCenter=ECOM-3200, Application=InventoryForecastAgent, Team=SupplyChainOps, Environment=Production.



קונסול ה-Projects של Mantle — שים לב לכיתוב bedrock-mantle endpoint בצד, ולעמודת התגים לכל פרויקט

שינוי הקוד כאן הוא הקל מבין השלושה: "השינוי קוד פה הם די מינורים" [13:49]. בלקוח של Anthropic מוסיפים default header עם ה-workspace ID; בלקוח של OpenAI מעבירים project ID. מרגע שהלקוח מאותחל כך, כל ה-inference שעובר דרכו יורש את התגים של אותו פרויקט או workspace.

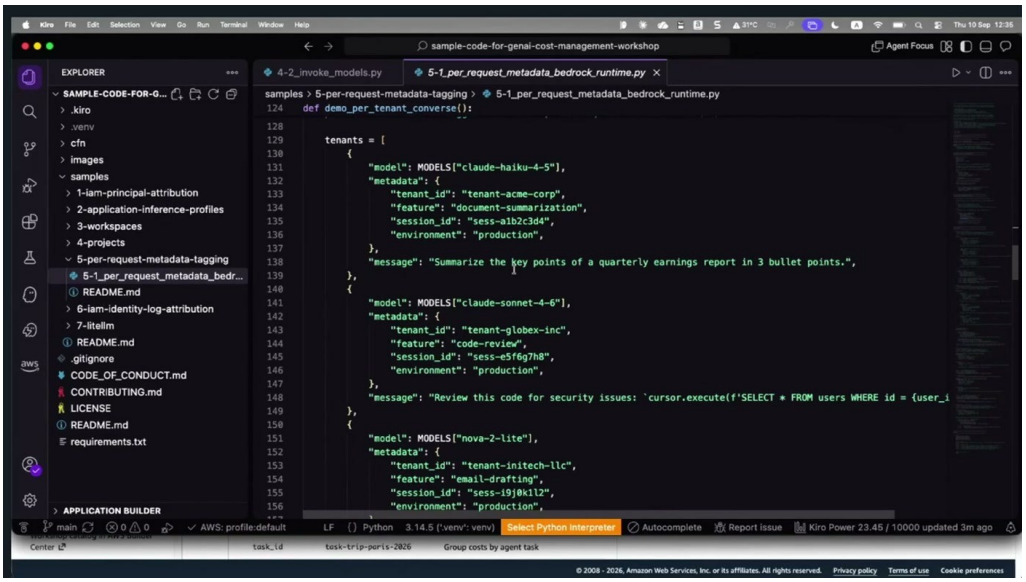
> **אותה מלכודת** גם כאן חלים שני מחזורי ההמתנה: 24 שעות לחלחול התגים, הפעלה ידנית שלהם כ-cost allocation tags, ואז המתנה נוספת עד שהנתון מופיע בבילינג [13:49].

8. שיטה 5 — Per-Request Metadata Tagging

המטרה: WHICH CALL — הרזולוציה הגבוהה ביותר. הפלט: ספירת טוקנים. העיכוב: כדקה. שינויי קוד: כן, בכל קריאה.

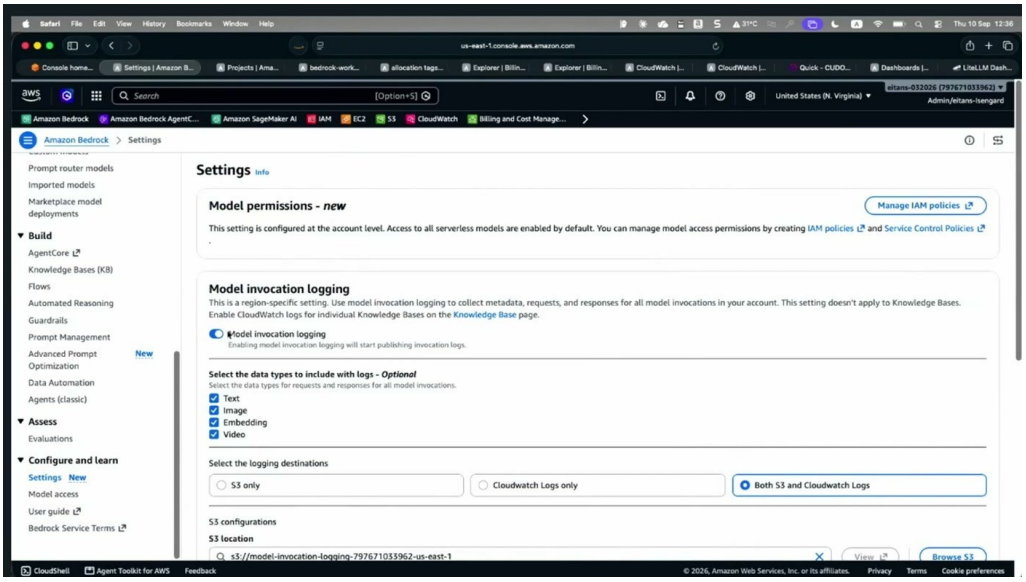
זו נקודת המפנה של הסשן, כי כאן משתנה לא רק הרזולוציה אלא גם צינור הנתונים. במקום שהתגים יגיעו לבילינג, הם נרשמים ל-Bedrock Model Invocation Log. התרחיש: חברת SaaS עם אלפי לקוחות, או ארגון גדול עם agent שאלפי משתמשים מפעילים, שרוצה לדעת פר-קריאה מי צרך מה.

מבחינת קוד, מוסיפים אובייקט metadata לקריאת Converse או InvokeModel. בדמו הודגמו שלושה — tenants tenant-acme-corp, tenant-globex-inc ו-tenant-initech-llc — כשכל אחד נושא tenant_id, feature, session_id ו-environment. המרצה ציין שאפשר להעביר עד 16 שדות של tag ו-[15:28] value.



קוד הדוגמה: אובייקט ה-metadata לכל tenant, ובעץ הקבצים משמאל — שבעת המודולים של הסדנה, אחד לכל שיטה

שים לב ללא הפעלת Model Invocation Logging השיטה הזאת לא מייצרת דבר: "Bedrock כברירת מחדל לא שומר את ה-input, ה-output והמטאדאטה של כלום, אלא אם אתם מאפשרים לו" [15:28]. זו הפעלה מודעת שיש לה גם משמעות רגולטורית — מרגע שהיא דולקת, ה-prompts וה-completions נשמרים ב-S3 או ב-CloudWatch.



מסך ההגדרות של Bedrock: מתג ה-Model invocation logging, בחירת סוגי הדאטה (Text, Image, Embedding, Video) ובחירת יעד — S3, CloudWatch או שניהם

מרגע שהלוג פעיל, התחקור נעשה ב-CloudWatch Logs Insights בשפת שאילתות, כשמפרקים את שדה ה-requestMetadata. המרצה הציג תוצאה שמכילה tenant ID, feature, model ID, total input tokens ו-request count.

[15:28] — ומה שטוב פה, בגלל שזה CloudWatch Logs או S3, זה ב-near real-time, זה בערך דקה כזה דילי

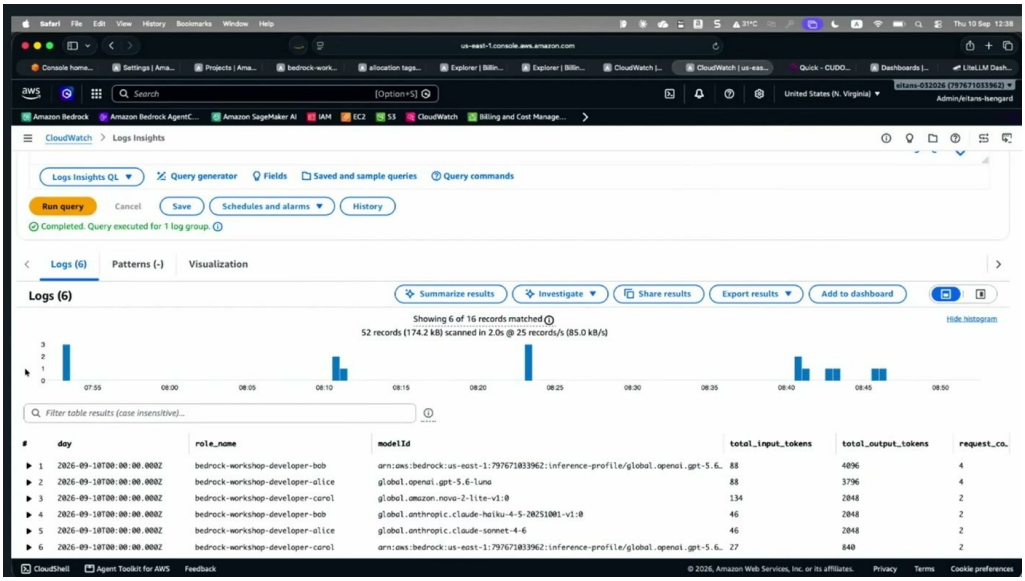
9. שיטה 6 — IAM Identity Log Attribution

המטרה: WHO, אבל ב-near-real-time. הפלט: ספירת טוקנים. העיכוב: כדקה. שינויי קוד: אפס.

זו התשובה הישירה למגבלת ה-24 שעות של שיטה 1, והמרצה הציג אותה ככזו במפורש. התובנה המכניסטית: Model Invocation Log שומר גם את ה-role של כל זהות שקוראת למודל — ולכן אפשר להפיק את אותו שיוך לפי מפתח, בלי להמתין לבילינג ובלי לגעת בקוד.

> **— [17:00]** אם יש לי עכשיו מפתחים ב-Claude Code שקוראים ל-Bedrock, אז ה-roles שלהם נשמרים ב-CloudWatch, מה שמאפשר לי בזמן אמת בעצם ללכת ל-CloudWatch... לתחקר את הלוגים האלה ולקבל תוצאות

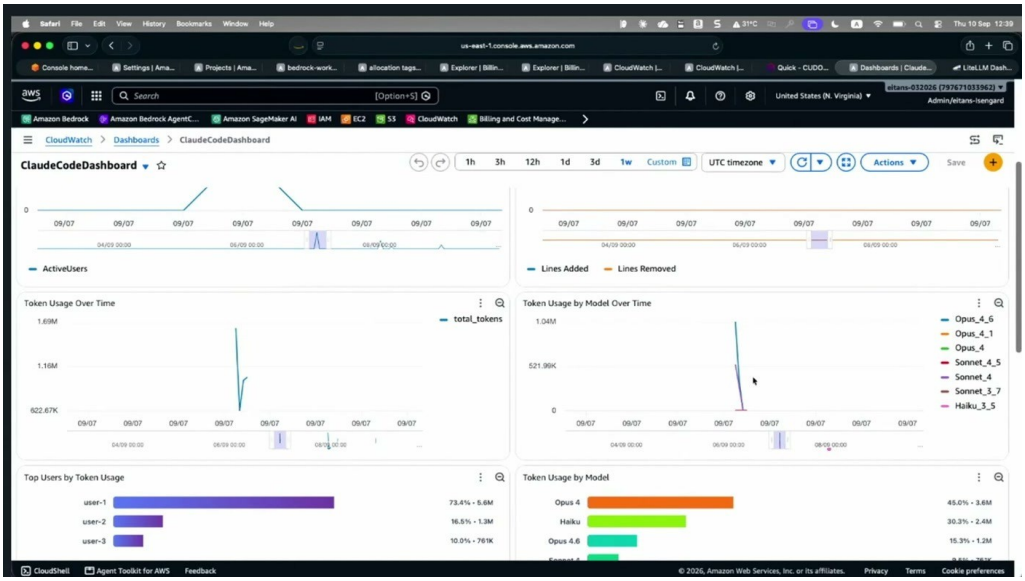
אותם שלושה מפתחי דמו — Alice, Bob ו-Carol — מופיעים כאן שוב, הפעם דרך שאילתת Logs Insights על שדה ה-role_name. התוצאה מראה פר-מפתח: באילו מודלים השתמש, כמה input tokens, כמה output tokens וכמה בקשות. בטבלה שהוצגה מופיעים מודלים שונים באותה שאילתה — global.openai.gpt-5.6, global.anthropic.claude-haiku-4-5, global.amazon.nova-2-lite ו-global.anthropic.claude-sonnet-4-6.



תוצאות שאילתת Logs Insights: שורה לכל צירוף של מפתח ומודל, עם ספירת טוקנים ובקשות

המרצה לא הריץ את הדמו בזמן אמת בשל אילוצי זמן, והודה בכך בפה מלא: "אני מניח שיש פה אנשים כזה שלא מאמינים לי — אם היה יותר זמן הייתי מריץ את הדמו, מחכה דקה ומראה לכם שזה באמת משתנה בדקה דילי. תצטרכו לסמוך עליי כאן" [18:35].

מעל השיטה הזאת נבנו כלים. הוצג dashboard פתוח ב-CloudWatch בשם ClaudeCodeDashboard, שמראה בזמן אמת משתמשים פעילים, שורות קוד שנוספו ונמחקו, צריכת טוקנים לאורך זמן, פילוח לפי מודל, ודירוג Top Users by Token Usage. "יש כל מיני פתרונות open source ב-AWS שבנינו בשביל לאפשר ללקוחות שלנו שמשתמשים ב-Claude Code או ב-Codex לעשות מעקב אחרי יוזרים בזמן אמת ולחסום אותם" [18:35].



הדשבורד הפתוח למעקב אחרי Claude Code: משתמשים פעילים, טוקנים לאורך זמן, פילוח לפי מודל ודירוג משתמשים מובילים

המרצה הוסיף גם שמתנהלת עבודה פנימית לצמצום הפער של השיטות מבוססות-הבילינג: "אנחנו עובדים ב-AWS פנימית על לשפר את זה, בשביל שזה יגיע יותר לממדים של near real time מבחינת בילינג" [17:00].

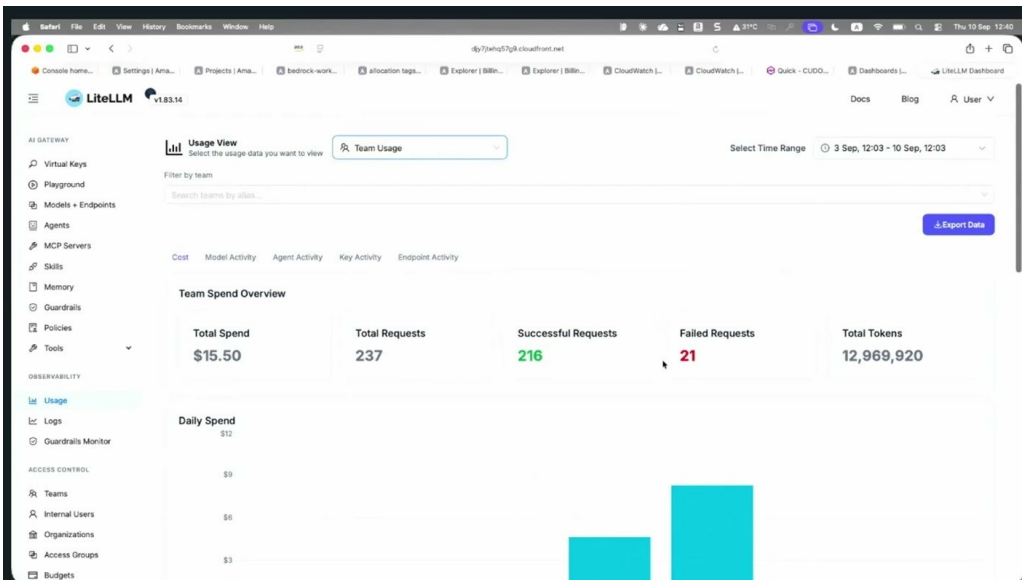
10. שיטה 7 — AI Gateway (LiteLLM ואחרים)

המטרה: WHO ו-WHAT, עם UI ואכיפת תקציבים. הפלט: אומדן עלות. העיכוב: real-time. שינויי קוד: הפניית ה-client ל-gateway.

השיטה היחידה שאיננה native הוצגה מתוך צורך מוגדר: הדשבורד הפתוח נותן ויזיביליות, אבל לא נותן ממשל. המרצה תיאר את הדרישה שהוא שומע מלקוחות — UI שבו מנהל צוות יכול להקצות תקציב יומי למפתח, ומנהל מחלקה יכול להגדיל אותו.

> **— [20:08]** עכשיו יש איזה צוות עם איזה מפתח שהגיע לתקציב שלו, אז איך ראש הצוות שלו יכול לתת לו עוד איזה כמה דולרים ליום, והמנהל מחלקה יכול לתת לו אולי 100 דולר ליום

הדמו רץ על LiteLLM, אך המרצה הדגיש שוב ושוב שאין כאן המלצה בלעדית וששני מוצרים נוספים בשימוש לקוחות True Foundry — AWS ו-Bifrost (האחרון open source עם רישיון מסחרי נלווה). ה-gateway מתחבר לכל מודלי Bedrock וגם לספקים אחרים, ומאפשר תצוגת usage פר-צוות.



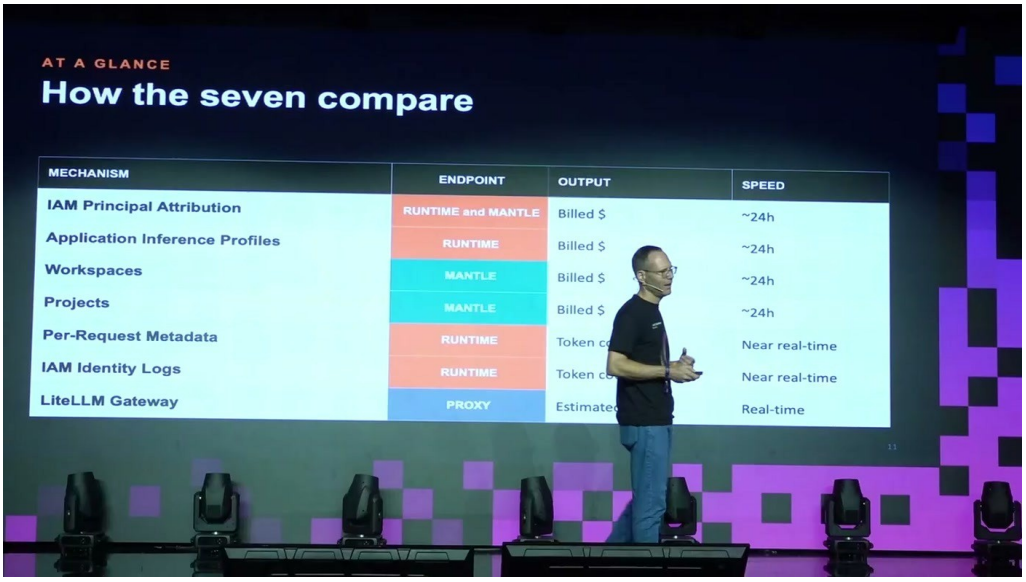
מסך ה-Team Usage של ה-gateway: סך הוצאה, מספר בקשות מוצלחות וכושלות וסך טוקנים לתקופת זמן נבחרת

מנגנון הממשל בנוי על שתי שכבות. ברמת הצוות מוגדר תקציב — בדמו צוות עם תקציב של 100 דולר שצרך עד כה 15.50 דולר. ברמת המפתח מנפיקים Virtual Keys, ולכל מפתח אפשר להגדיר תקציב ואילו מודלים מותרים לו. הדוגמה שהוצגה: מפתח junior עם תקציב של 7 דולר ומודלים פשוטים, מול מפתח senior עם תקציב בלתי מוגבל וגישה לכל המודלים [21:40]. כשמפתח מגיע ללימיט, מנהל נכנס להגדרות ומוסיף לו סכום להמשך היום.

שתי הערות תפעוליות שנאמרו: בגרסה המסחרית קיים SSO ארגוני, ואז אין צורך להנפיק Virtual Keys ידנית; ו-LiteLLM תומך גם בתגים, כך שאפשר לבצע דרכו per-request metadata tagging — אבל "זה יישמר ב-LiteLLM, בדאטאבייס שלו, לא ב-AWS" [20:08].

11. ההשוואה המרוכזת

הסליד שסגר את חלק הדמואים הוא המסמך הישים ביותר בסשן — טבלה אחת שמכריעה איזו שיטה מתאימה לאיזה צורך, לפי endpoint, סוג פלט ומהירות.



שבע המכניקות זו מול זו — עמודת ה-endpoint מבחינה בין RUNTIME ל-MANTLE, ועמודת המהירות בין 24 שעות ל-near real-time

מכניקה	Endpoint	פלט	מהירות
IAM Principal Attribution	Runtime ו-Mantle	$ Billed	24h~
Application Inference Profiles	Runtime	$ Billed	24h~
Workspaces	Mantle	$ Billed	24h~
Projects	Mantle	$ Billed	24h~
Per-Request Metadata	Runtime	Token count	Near real-time
IAM Identity Logs	Runtime	Token count	Near real-time
LiteLLM Gateway	Proxy	$ Estimated	Real-time

המרצה סיכם את הטבלה בשלוש הכרעות פשוטות: לשיוך לפי זהות — IAM Principal Attribution; לשיוך לפי אפליקציה או Application Inference Profiles — agent ב-runtime, או Workspaces ו-Projects ב-Mantle; לשיוך פר-prompt או פר-קריאה — [23:16] Request Metadata Tagging.

> **הפער שנשאר פתוח** המחיר של השיטות ה-near-real-time נאמר במפורש: הן מודדות טוקנים, לא דולרים. "אפשר לחסום יוזרים, אבל כן צריך לזכור שזה בטוקנים, וצריך לעשות את החישוב של טוקן כפול מחיר של מודל בשביל ממש לקבל תקציב" [21:40]. כלומר טבלת מחירי מודלים מתוחזקת היא תנאי לאכיפת תקציב בזמן אמת.

12. מוויזיביליות לחיסכון

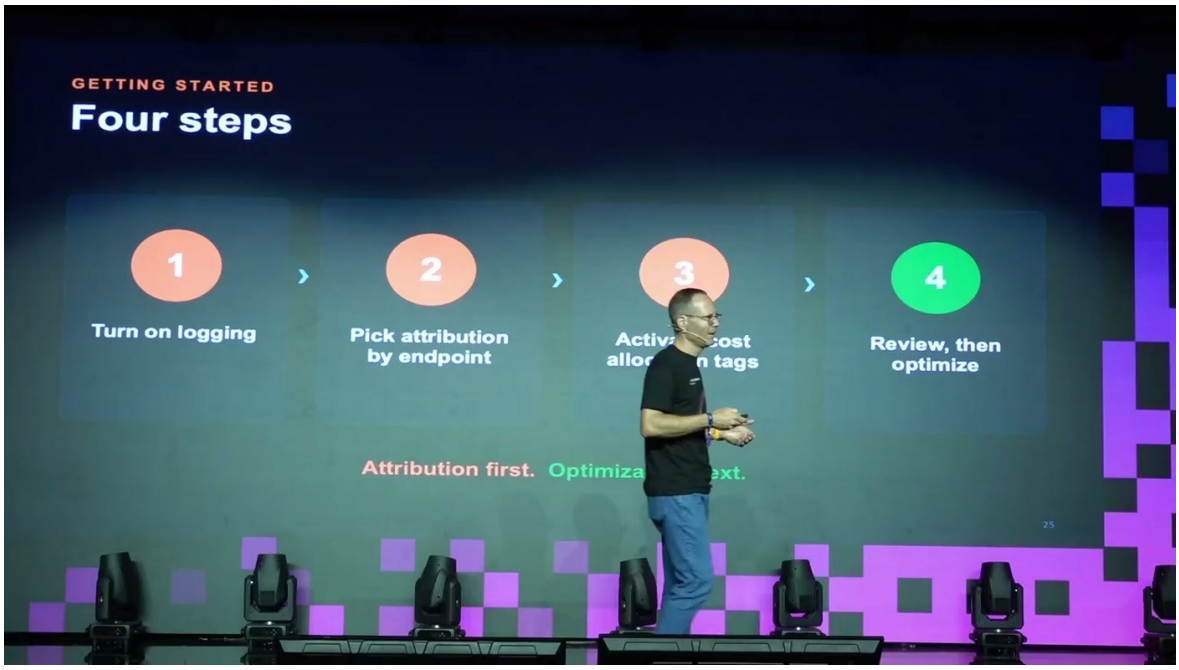
החלק האחרון של הסשן נגע רק בקצרה באופטימיזציה, ובכוונה. הטענה: הצוואר האמיתי הוא הזיהוי, לא הטיפול.

> **[23:16] —** וזה באמת האתגר, כי קודם כל קשה לשים את האצבע איפה העלות הולכת — וזה האתגר המרכזי. ברגע שכבר יודעים איפה העלות, על מה העלות מתבזבזת ומי הצרכן... יש המון דברים שאפשר לעשות

שתי הדוגמאות שניתנו לצרכן החריג הן בדיוק המקרים שמייצרים חשבונית מפתיעה: "זה יכול להיות מפתח שעכשיו הגיע לאיזה לופ אין-סופי ומבזבז את התקציב מאוד גבוה ב-Claude Code, או סוכן שהלך לכיוונים לא טובים ומבזבז יותר מהרגיל" [23:16]. שניהם מתגלים רק אם יש שיוך — ואם השיוך near-real-time, גם ניתנים לעצירה.

רשימת האופטימיזציות הוצגה כרשימת כותרות בלבד, עם הסתייגות מפורשת: "כל מה שאמרנו פה בשקף הזה זה נושא לסשן שלם" [24:48]. שלוש שהוזכרו בדיבור: בחירת מודל מתאים למשימה, prompt caching, ו-cross-region inference — שהוצג כמוזיל עלויות וגם משפר ביצועים.

13. מה לוקחים מכאן



סדר הפעולות להתחלה, וההיררכיה שסגרה את הסשן: קודם שיוך, אחר כך אופטימיזציה

- **1. להדליק logging.** Model Invocation Logging ב-Bedrock הוא תנאי מקדים לכל השיטות ה-near-real-time, והוא כבוי כברירת מחדל. זו גם החלטה שיש לה משמעות רגולטורית, כי ממנה והלאה נשמרים prompts ו-completions.
- **2. לבחור שיטת שיוך לפי ה-endpoint.** ההכרעה הראשונה היא טכנית ולא עסקית: runtime או Mantle. היא פוסלת מראש חלק מהשיטות.
- **3. להפעיל cost allocation tags.** השלב שהכי קל לשכוח — תגים חדשים מופיעים כ-Inactive ודורשים הפעלה ידנית, ואחריה המתנה נוספת.
- **4. לבדוק, ורק אז לייעל.** "Attribution first. Optimization next" — אין טעם לבחור מודל זול יותר לפני שיודעים מי צורך.
- **להפריד בין שני עולמות המדידה.** בילינג נותן דולרים אמיתיים בעיכוב יום; לוגים נותנים טוקנים בעיכוב דקה. לתקציבים ולחסימה בזמן אמת — רק המסלול השני, ורק עם טבלת מחירי מודלים משלכם.
- **השיטות אינן חלופיות.** הסליד שקדם לסיכום הציג "Combine the lenses" — WHO ועוד WHAT ועוד WHICH CALL כשלוש זוויות על אותו workload, לא כבחירה ביניהן.
- **הסדנה זמינה לעבודה עצמאית.** כל מה שהוצג בסשן מפורק לשבעה מודולים בסדנה הרשמית Track and Optimize Generative AI Spend on AWS, כולל ריפוזיטורי קוד — "אפשר לסרוק את זה, לעשות את זה לבד במשרד או בבית" [24:48].

14. מילון מונחים

מונח	משמעות כפי שהוצגה בסשן
Mantle	inference engine חדש של Bedrock, תואם ל-SDK של OpenAI ושל Anthropic. מספיק להחליף Base URL. מחזיק את מושגי Projects ו-Workspaces.
Application Inference Profile	אובייקט שיוצרים מעל מודל ב-Bedrock runtime, נושא תגים משלו ומשמש כ-modelId בקריאה. המנגנון לשיוך פר-אפליקציה או פר-agent.
Project / Workspace	המקבילה ב-Mantle לאותו רעיון — Project מגיע מ-OpenAI, Workspace מ-Anthropic. מזוהים ב-client ונושאים תגים.
Cost Allocation Tag	תג AWS שהופעל ידנית לשימוש בבילינג. תגים חדשים מגיעים כ-Inactive; ההפעלה והחלחול לוקחים עד 24 שעות.
Model Invocation Log	לוג של Bedrock, כבוי כברירת מחדל, ששומר בקשות, תשובות ומטאדאטה ל-S3 ו/או ל-CloudWatch. הבסיס לשתי השיטות ה-near-real-time.
Per-Request Metadata	אובייקט JSON (עד 16 שדות) שמצורף לקריאת Converse או InvokeModel ונרשם ללוג. מאפשר שיוך ברמת tenant, feature או session.
IAM Identity Log Attribution	תחקור ה-role_name בתוך Model Invocation Log כדי לקבל שיוך פר-מפתח בעיכוב של כדקה, בלי שינוי קוד.
AI Gateway	שכבת proxy מול המודלים שמספקת ממשל, תקציבים ו-UI. הוצגו LiteLLM (בדמו), True Foundry ו-Bifrost.
Virtual Key	מפתח אישי שמנפיקים ב-gateway למפתח בודד, ואליו מצמידים תקציב ורשימת מודלים מותרים.
Converse API	ה-API האחיד של Bedrock לשיחה עם מודלים. נשאר ללא שינוי ברוב השיטות; רק ה-modelId או ה-metadata משתנים.

להמשך הסשן נסגר בהפניה לסדנה הרשמית בקטלוג AWS Workshops, catalog.workshops.aws/track-genai-spend-on-aws, המכילה מודול נפרד לכל אחת משבע השיטות עם צילומי מסך וקוד לדוגמה.