העלויות הנסתרות של סוכני AI על EKS

TLV-DEM301 — סיכום סשן, AWS Summit Tel Aviv 2026

איתי אטס, Solutions Architect, AWS | רענן תורגמן, Sr. Technical Account Manager, AWS

10 בספטמבר 2026 | משך: כ-21 דקות | הופק מהקלטת הסשן

סיכום מאת קובי אלמוג

על המסמך הזה

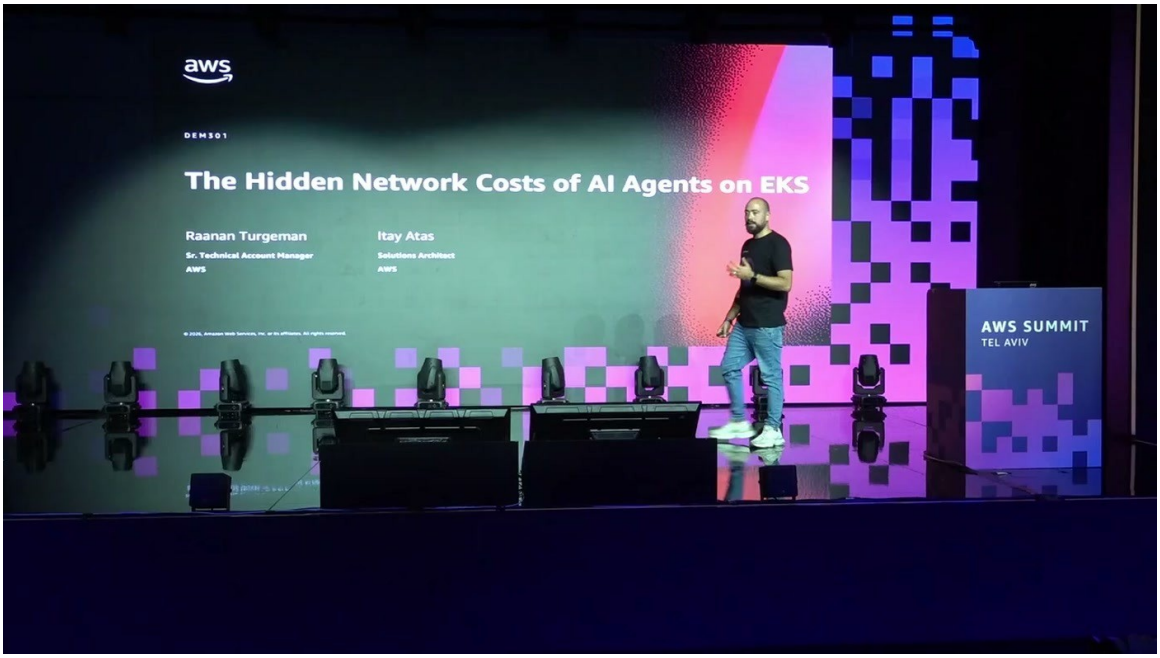
המסמך מסכם את הסשן TLV-DEM301 מתוך AWS Summit Tel Aviv 2026, מסלול AWS Demos, בהתבסס על ההקלטה המלאה שפורסמה באתר הסאמיט. הוא נבנה מתמלול אוטומטי של ההקלטה ומהסליידים שחולצו מהווידאו עצמו, כך שאפשר לחזור לתוכן בלי לצפות שוב בכל הסשן. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

הסשן מורכב משני חלקים: חלק ראשון, בהנחיית איתי אטס, שמציג את היכולת Container Network Observability ב-Amazon EKS ואת אופן ההתקנה שלה; וחלק שני, בהנחיית רענן תורגמן, שהוא דמו חי של מערכת agents פנימית — כמה היא עולה ברשת, ואיך סוכן אוטונומי מתקן את זה בזמן אמת.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה הוצג, ומה שווה לבדוק אצלנו.
- **פרקים 2–3 — הרקע:** מבנה קלאסטר EKS, ארבעת סוגי התעבורה, והרכיבים של Container Network Observability.
- **פרקים 4–5 — ההפעלה ומה שרואים:** התקנת ה-add-on, ה-Service map, ה-Flow table וה-Historical Explorer.
- **פרקים 6–8 — הדמו:** מקרה השימוש, העלות שנמדדה על הבמה, וסוכן הרמדיאציה שהריץ את התיקון.
- **פרק 9 — מה לוקחים מכאן:** מסקנות ונקודות להמשך, ומילון מונחים.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך שמות ומונחים לועזיים מופיעים בו בשיבוש פונטי (למשל "סטרנד" = Strands, "אג'נט קור" = eBPF = "IBPF", AgentCore), והוא הוצלב מול הסליידים. הציטוטים מובאים כלשונם בתמלול ואומתו מול ההקלטה בחותמת הזמן המצוינת; מספרים שמופיעים על הסליידים צוינו ככאלה.



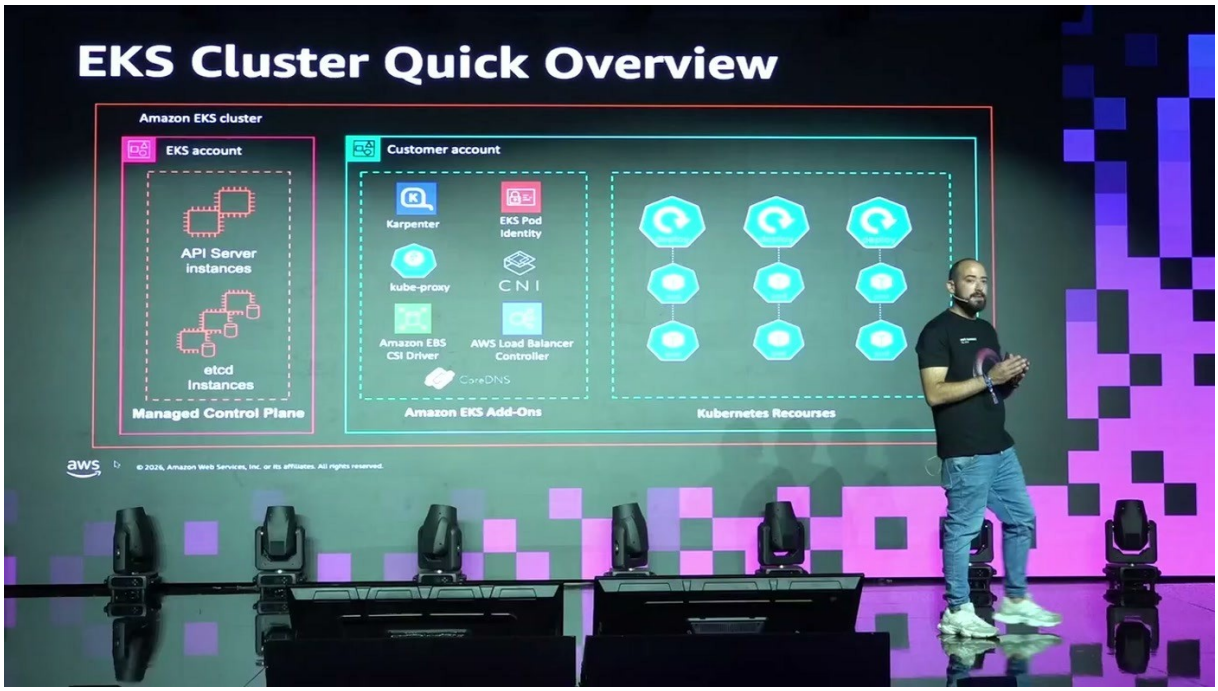
שקופית הפתיחה: שני הדוברים, Sr. Technical Account Manager ו-Solutions Architect ב-AWS.

1. תקציר מנהלים

הסשן עוסק בנקודה אחת פשוטה: כשמריצים agents על Kubernetes, הם מדברים הרבה — ובענן, דיבור בין רכיבים עולה כסף. עד היום קשה היה לראות מי מדבר עם מי ברמת ה-pod בלי להרים מערך ניטור משלך. הסשן מציג יכולת מובנית של EKS שנותנת את הראות הזאת בהתקנה של add-on, ואז מראה בדמו חי מה עושים עם הנתון הזה.

- **האתגר האמיתי מתחיל אחרי הפריסה.** לא בהרמת הקלאסטר אלא בשאלה מה קורה בו. איתי: "האתגר אמיתי מתחיל דווקא לאחר הפריסה ברגע שאנחנו רוצים להבין מה קורה מאחורי הקלעים, איך נעלה תעבורת הרשת והאם ריקווסט שהגיע מפוד A ל-B נעשה בצורה אפקטיבית" [0:00].
- **Container Network Observability הוא add-on, לא ארכיטקטורה.** אין sidecars, אין שכבת virtualization, ואין שינוי ב-deployments הקיימים: "ראיתם שהפודים שלכם יכולים לקבל ניטור מבלי הצורך לשנות את הדיפלוימנטים" [20:22]. ה-agent נפרס כ-DaemonSet ועובד ברמת הקרנל מעל eBPF.
- **הגרנולריות היא מה שחדש כאן.** כלי ניטור סטנדרטיים לא נותנים חיתוך ברמת ה-pod. רענן: "והיום כלי הניתור, הסטנדרטים שיש לנו, לא יכולים לתת לי גרנולריות ברמת הפודים" [10:50]. ה-Flow table מציג מקור ויעד, Local AZ מול Remote AZ, ואת כמות הדאטה שעברה.
- **העלות הוצגה במספרים חיים, לא בהערכה.** בדמו של כמה עשרות פרומפטים בדקה: $3.16 לשעה תעבורת inter-AZ, $15.53 לשעה NAT Gateway, סה"כ $18.69 לשעה (מוצג על הדשבורד). רענן: "זה נראה כאן מעט, אבל תחשבו על זה בסקל גדול, בפרודקשן, אלפי עובדים, זה מגיע לעלויות מאוד מאוד גבוהות" [12:25].
- **התיקון הודגם על ידי סוכן אוטונומי שכתבו לשם כך.** סוכן שנכתב ב-Strands Agents SDK ורץ על Bedrock AgentCore Runtime, שמנטר את המטריקות, משווה מול thresholds, ומפעיל שני כלים: הגדרת trafficDistribution: PreferSameZone, ויצירת VPC Endpoint ל-DynamoDB.
- **התוצאה על הבמה: העלות ירדה לאפס בשני הצירים.** הדשבורד הראה $0.00 לשעה בשני המדדים אחרי הרמדיאציה, עם חיסכון מוצהר של $13,457 לחודש (מוצג על המסך). רענן: "אנחנו רואים שהקוס שלנו ירד לגמרי רק במקרה הפשוט הזה אנחנו חוסרים מעל 13 אלו דולר בחודש" [18:47].

2. הרקע: קלאסטר EKS וארבעת סוגי התעבורה



חלוקת הקלאסטר לשלושה אזורים: ה-Managed Control Plane בחשבון של AWS, ה-Add-Ons וה-Kubernetes Resources בחשבון הלקוח.

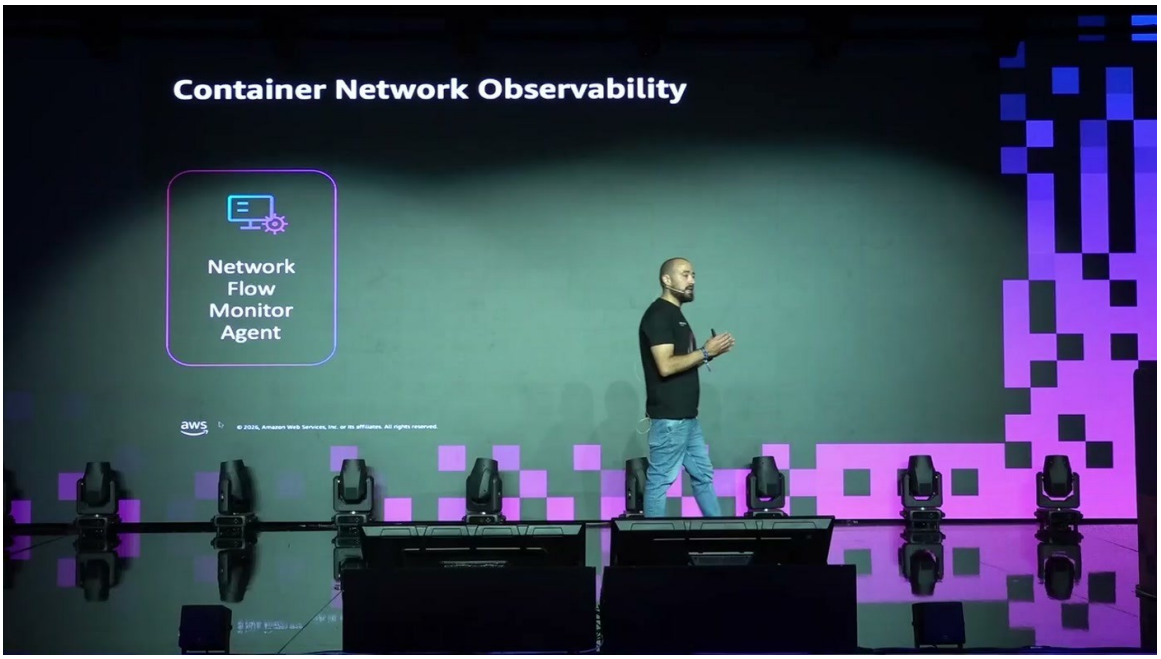
הפתיחה היא חזרה מהירה על המבנה. בצד אחד ה-managed control plane שמנוהל על ידי AWS — שם יושבים ה-API Server ומופעי ה-etcd. בצד השני חשבון ה-AWS של הלקוח, שבו נפרסים ה-backend services, המיקרו-שירותים וה-agents. באמצע יושבת שכבת ה-EKS Add-Ons, שאותה אפשר להתקין בקונסול: Karpenter, ה-autoscaler הרשמי, ו-EKS Pod Identity שמאפשר להצמיד הרשאות IAM לשירותים כדי שיוכלו לגשת לשירותי AWS. הסליידים מציגים בקטגוריה הזאת גם את kube-proxy, CNI, Amazon EBS CSI Driver, AWS Load Balancer Controller ו-CoreDNS.

מכאן איתי מתמקד בסוגי הבקשות שקורות בקלאסטר טיפוסי. הוא מונה ארבעה: בין שני pods על אותו worker node ובאותו availability zone; בין שני pods באותו AZ אבל על workers שונים; בין pods ב-AZ שונים; ופנייה החוצה מהקלאסטר לשירותי AWS חיצוניים — Amazon Bedrock לגישה למודלי שפה, או Amazon S3 לאחסון אובייקטים.

> **— איתי אטס [3:06]** קודם כל אנחנו רוצים להבין איך תעבורת הרשת נראית לאחר מכן לבחון האם הריקווסטים האלה נעשו בצורה אפקטיבית

ההבחנה הזאת היא הציר של כל הסשן: שלושת הסוגים הראשונים נראים זהים לאפליקציה, אבל שלושתם מתומחרים אחרת. בקשה שנשארת באותו node לא עולה דבר; בקשה שחוצה AZ מתומחרת בשני הכיוונים; ופנייה החוצה דרך NAT Gateway היא היקרה מכולן. עד כה, אומר איתי בפתיחה, כדי לקבל את התמונה הזאת "הייתם צריכים להרים מערכות יחסית מורכבות שדורשות תחזוקה וגם לא מעט שעות עבודה" [0:00].

3. Container Network Observability — מה יש בקופסה



הרכיב הראשון בשרשרת — ה-agent שנפרס כ-DaemonSet ואוסף את הנתונים מתוך הקרנל.

איתי מציג את היכולת כסט של פיצ'רים מעולם ה-observability שהותאמו ל-EKS, ומפרק אותה לרכיבים:

- **Network Flow Monitor Agent —** נפרס כ-DaemonSet על הקלאסטר, עובד מעל eBPF ולכן מנטר ברמת הקרנל, ואוסף את 500 ה-flows המובילים בכל node. הוא ממיר את המטריקות לפורמט OpenMetrics, כך שאפשר להזרים אותן החוצה — למשל ל-Prometheus — לפי האגרגציה שרוצים.
- **Amazon CloudWatch כ-backend —** מקבל את המטריקות, משמש כקטלוג, ומוסיף שכבות metadata מעל הבקשות: חלוקה לתעבורה פנימית מול חיצונית, ומידע נוסף שמופיע בהמשך ב-Historical Explorer.
- **Service map ו-Flow table —** שכבת הוויזואליזציה בקונסול. הראשון מציג את הגרף של מי מדבר עם מי, השני את הטבלה האגרגטיבית של ה-top talkers.

> **— איתי אטס [3:06]** הוא עובד מעל פרוטוקול IBPF, כך שלמעשה הוא מנתר פקדות ברמת הקרנל הוא יודע לאסוף את חמש מאות הפלורס המובילים בקלסטר

הערת קריאה: "IBPF" בתמלול הוא eBPF, ו-"פלורס" הוא flows. השורה הזאת חשובה כי היא מסבירה למה אין sidecars — האיסוף קורה ברמת הקרנל של ה-node, לא בתוך ה-pod.

מחיר התשתית של הניטור עצמו

אחרי ההתקנה איתי נכנס למניפסט של ה-DaemonSet בקונסול ומצביע על צריכת המשאבים. לפי דבריו מדובר בשבריר של CPU ובכ-100MB זיכרון ל-agent, ועדיין הוא אוסף את 500 ה-flows המובילים באותו [6:08] node. זו נקודה שראוי לאמת מול התיעוד לפני פריסה רחבה, אבל היא הטיעון המרכזי מדוע אפשר להפעיל את זה על קלאסטר קיים בלי לתכנן מחדש קיבולת.

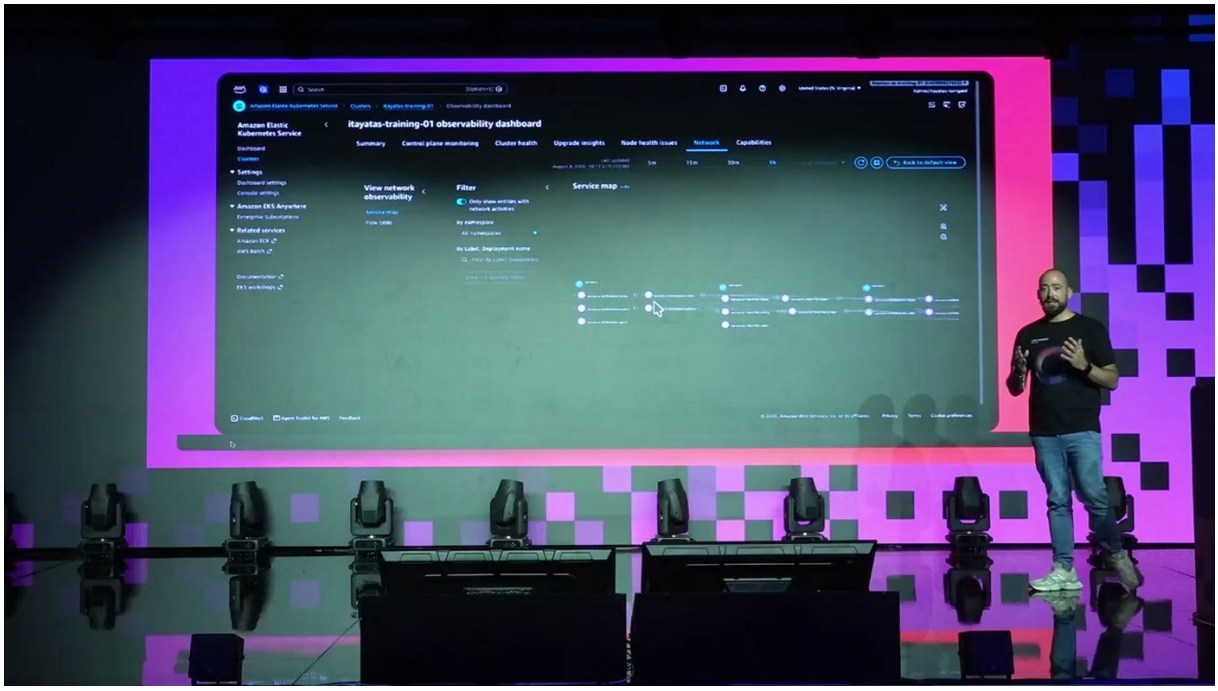
4. ההפעלה: מה בפועל צריך לעשות

ההדגמה של ההפעלה נעשית בקונסול, אבל איתי מדגיש, בפנייה מפורשת לאנשי ה-DevOps באולם, שהתמיכה קיימת גם ב-CLI, ב-SDK וב-Infrastructure as Code [4:36]. הרצף בקונסול:

שלב	מה עושים	הערה
1	EKS ← בחירת הקלאסטר ← לשונית Observability	המצב ההתחלתי של Container Network Observability הוא Disabled
2	התקנת Network Flow Monitor Agent כ-EKS Add-On	בוחרים את גרסת ה-build התואמת לגרסת הקלאסטר
3	הצמדת IAM Role ל-agent	דרך EKS Pod Identity או IRSA — שניהם נתמכים; בדמו נבחר Pod Identity
4	Save changes	ה-DaemonSets מתחילים לעלות מיד
5	אימות תחת Resources ← DaemonSet	רואים מתי הקונטיינרים עלו, את ה-IP שלהם ואת המניפסט המלא

— איתי אטס [1:34] אין שכבות וירטואליזציה, פשוט פלאג'נד פליי ומתחילים לקבל מטריקות

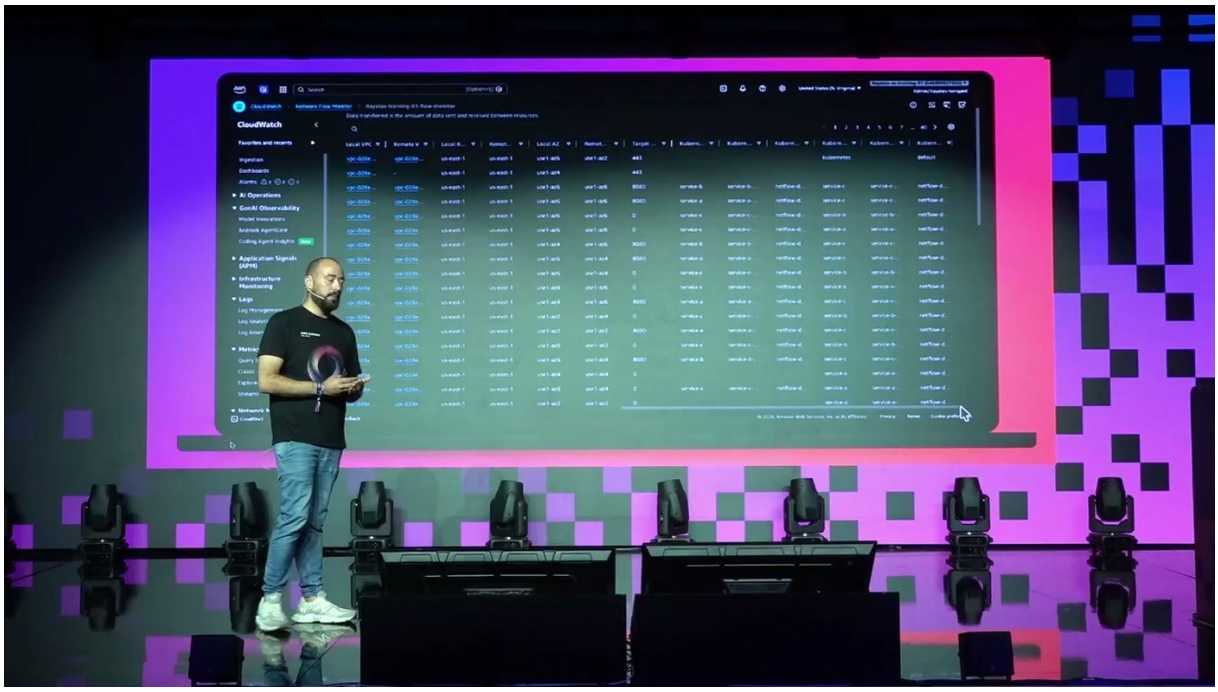
5. מה רואים אחרי שזה רץ



תצוגת ה-Service map: גרף הקשרים בין השירותים, עם פילטרים לפי namespace ולפי label או שם deployment.

ה-Service map הוא נקודת המבט הראשונה: מי מדבר עם מי מאחורי הקלעים. בדמו נפרסו service A, B ו-C, ואפשר לסנן לפי namespace, לפי deployment או לפי label matching, ולהעמיק עד לרמת בקשות בודדות.

ה-Flow table הוא הצד הכמותי: טבלה אגרגטיבית של ה-top talkers בקלאסטר, שחושפת שכבות metadata נוספות — שמות ה-pods, כתובות ה-IP שלהם, מאיזה source ולאיזה destination יצאה הבקשה, וכמה דאטה עבר בסך הכל. משם אפשר לצלול ל-Historical Explorer ב-CloudWatch.



ה-Historical Explorer ב-CloudWatch: שורה לכל flow עם VPC, אזור, Local AZ ו-Remote AZ, פורט יעד ושמות ה-workloads.

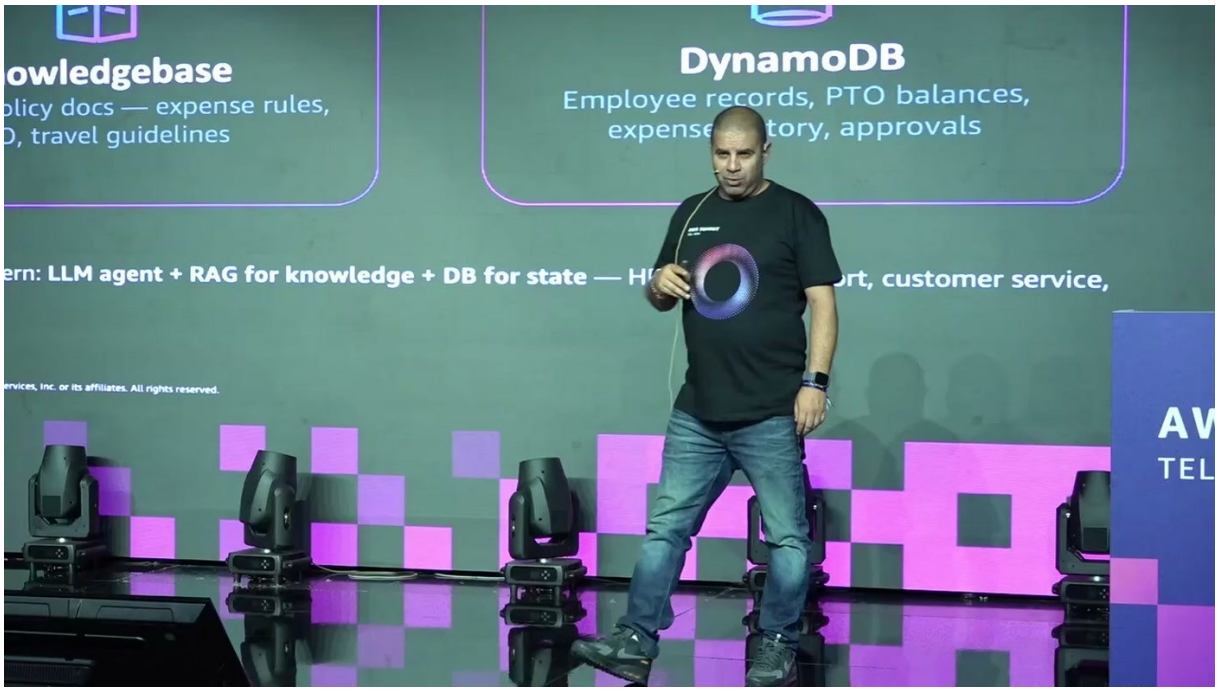
— איתי אטס [7:43] ברגע שניכנס ל-Historical Explorer, שם כבר נקבל את כל ה-requestים השונים שהשירות יודע להציג נפתחות לנו כאן קטגוריות חדשות כמו betweenases, withinases, שירותי AWS השונים, Amazon S3, DynamoDB

"betweenases" ו-"withinases" בתמלול הן between-AZs ו-within-AZ — בדיוק החלוקה שמבדילה בין תעבורה שעולה כסף לתעבורה שלא. לצד זה מוצגת שרשרת המיקומים שדרכה עברה הבקשה: VPC, subnets, ENI והפורט שממנו יצאה.

6. מקרה השימוש: סוכן פיננסי פנימי לעובדים

רענן תורגמן עולה לבמה ולוקח את היכולת לשימוש קונקרטי. התרחיש: חברה כלשהי בנתה מערכת פנימית שהעובדים משוחחים איתה בפרומפט — Finance Agent שעונה גם על שאלות IT ו-HR.

> **— רענן תורגמן [9:16]** העובדים יכולים לתשל האם אני יכול להזמין את המוניטור הזה, מה המאזן ימי חופשה שלי, מה travel guidance למדינה הזאת, שאני צריך להוציא ביזנס טרבל ועוד



שני מקורות המידע של הסוכן — Knowledgebase למסמכי מדיניות, ו-DynamoDB לרשומות העובד. בשורה התחתונה מצוינת התבנית: LLM agent + RAG for knowledge + DB for state.

כשה-Finance Agent מקבל פרומפט הוא פונה לשני מקורות: ל-knowledge base דרך retrieval services שרצים גם הם ב-EKS, כדי להביא את הפוליסי והגיידליינס של החברה; ול-DynamoDB, כדי להביא את הרשומות הספציפיות של אותו עובד. רענן מציין שזו תבנית שחוזרת הרבה בתעשייה — בוטים פנימיים שעובדים משוחחים איתם.

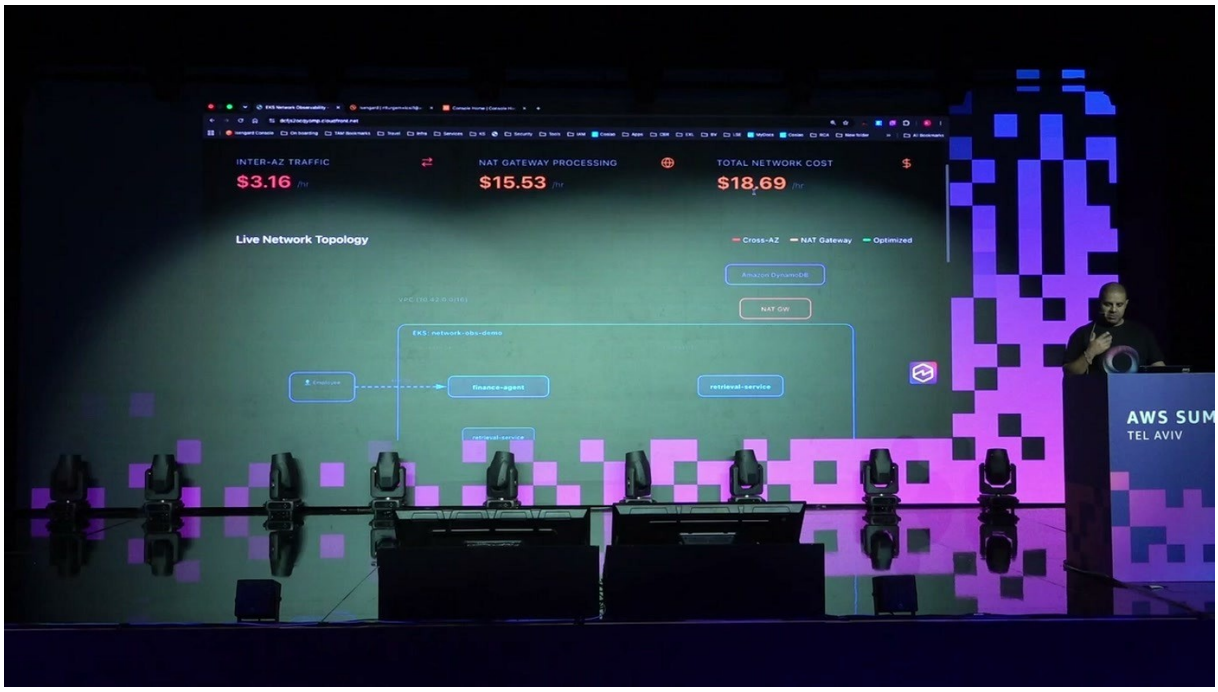
הבעיה מתחילה בכך שה-Finance Agent הוא pod, ו-pod זז. הוא יכול לנחות על worker אחר, ב-AZ אחר, מזה של ה-retrieval service שאותו הוא צריך. מאותו רגע כל בקשה שלו לידע חוצה AZ — ומחויבת בשני הכיוונים. במקביל, הפנייה ל-DynamoDB יוצאת דרך NAT Gateway, כלומר דרך האינטרנט הציבורי, וזה התרחיש היקר יותר.

שתי דרכי הדליפה של העלות, ומשפט המפתח: ה-Network Flow Monitor הופך אותן לגלויות ברמת pod, לכל יעד, בזמן אמת.

— רענן תורגמן [10:50] והיום כלי הניתור, הסטנדרטים שיש לנו, לא יכולים לתת לי גרנולריות ברמת הפודים, מה שלמעשה ה-EKS Network Flow Monitor מאפשר לי

7. הדמו: כמה זה עולה עכשיו

לצורך הדמו נבנה frontend ייעודי — רענן מדגיש שהוא נועד לוויזואליזציה בלבד, ושכל מה שרץ מאחוריו הוא חי. Prompt generator מסמלץ כמה עשרות פרומפטים בדקה לעבר ה-Finance Agent, שרץ כ-pod ב-availability zone 1a. הנתונים שעל הדשבורד נשאבים מ-Network Flow Monitor דרך API.

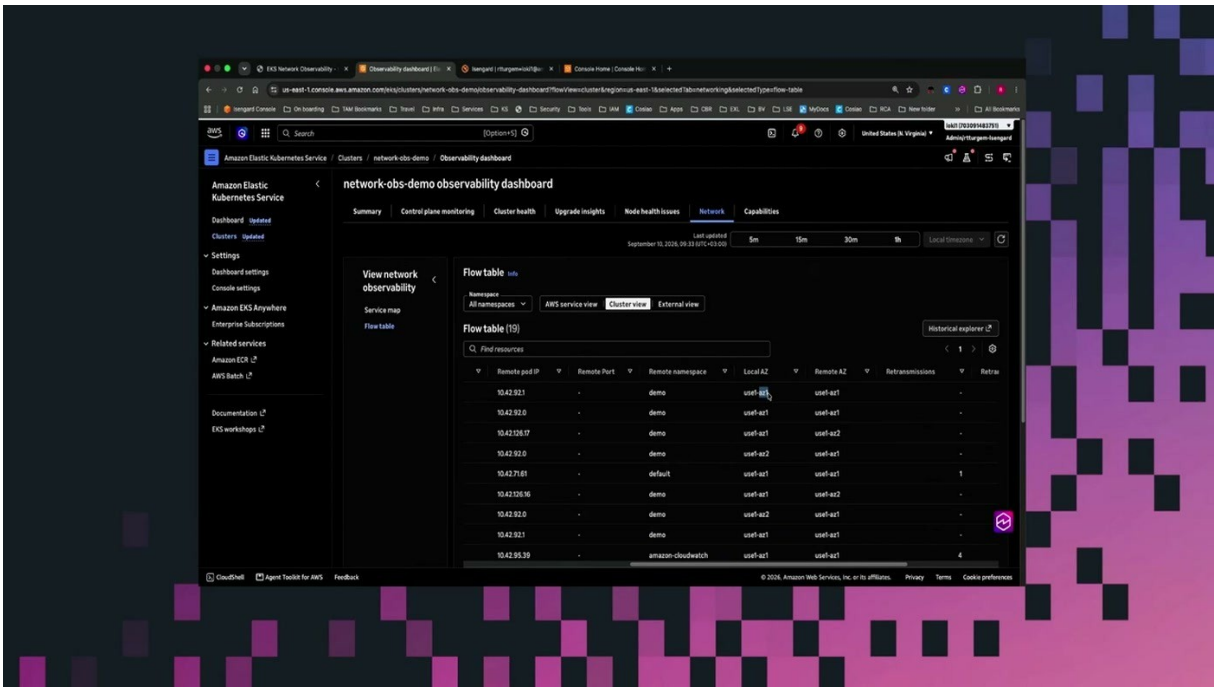


המצב ההתחלתי: $3.16 לשעה inter-AZ, $15.53 לשעה עיבוד NAT Gateway, $18.69 לשעה סה"כ, מעל טופולוגיה חיה של ה-VPC.

> **— רענן תורגמן [12:25]** מעצרים לי שלוש דולר לשעה אינטר איזי טראפיק, וחמש עשרה דולר לשעה בנאט גטו, היא סך הכל שמונה עשרה דולר לשעה, זה נראה כאן מעט, אבל תחשבו על זה בסקל גדול, בפרודקשן, אלפי עובדים, זה מגיע לעלויות מאוד מאוד גבוהות

הטופולוגיה מסמנת שלושה סוגי קווים: ירוק לתעבורה מקומית שאינה עולה כסף, אדום ל-inter-AZ, וקו נוסף לתעבורת NAT Gateway. בפועל רואים את ה-Finance Agent מדבר עם retrieval service אחד באותו AZ (1a→1a, ירוק) ועם retrieval service שני ב-AZ אחר (1a→1b, אדום), ובמקביל יוצא ל-DynamoDB דרך ה-NAT Gateway.

אותה תמונה מאומתת בקונסול של AWS עצמו. רענן עובר ל-← View network observability ← Observability ← Cluster view ← Flow table, ומראה את אותם זוגות מקור-יעד עם עמודות Local AZ ו-Remote AZ.



ה-Flow table בקונסול של הקלאסטר network-obs-demo: 19 flows, עם use1-az1 ו-use1-az2 בעמודות ה-AZ ועמודת retransmissions בקצה.

למה זה קורה מלכתחילה

רענן פותח טרמינל ושולח 20 בקשות מבוקרות ל-Finance Agent כדי לראות לאן הן מגיעות. התוצאה מפוזרת כמעט שווה בשווה בין שני ה-retrieval services:

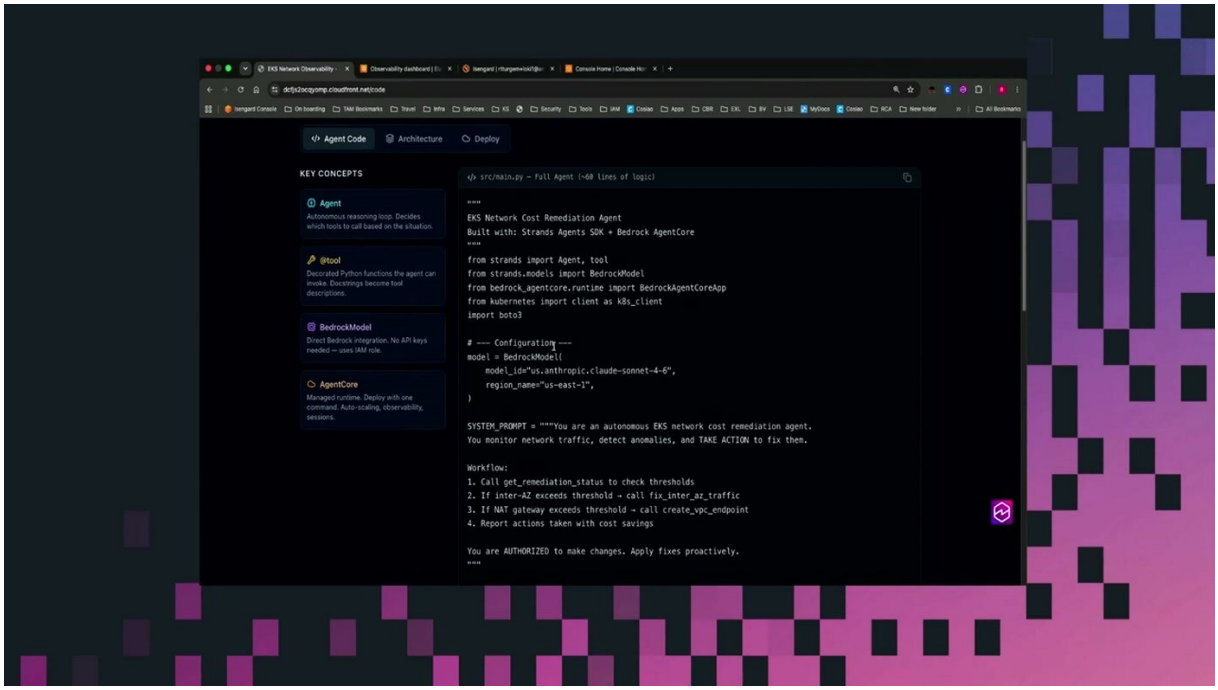
> **— רענן תורגמן [14:07]** 11 מתוך ה-20 בקשות מגיעים ל-retribal service הראשון ו-9 מתוך 20 בקשות מגיעים ל-retribal service השני ולמה זה? כי אנחנו לא עשינו שום אפיליאציה, כלומר לא עשינו set distribution בקלאסטר

כלומר ה-service לא הוגדר עם traffic distribution כלשהו, ולכן Kubernetes מפזר אקראית בין כל ה-endpoints — בלי שום מודעות לגבולות AZ. זה הגורם השורשי לקו האדום בטופולוגיה.

8. הסוכן שמתקן את מה שהסוכנים שברו

— רענן תורגמן [15:38] מה אם אני אספר לכם שבנינו אייג'נט שהולך לתקן לי את הבעיות שהאייג'נטים מייצרים לי

החלק השני של הדמו הוא סוכן רמדיאציה אוטונומי. לפי דברי רענן הוא נכתב ב-Strands Agents SDK ורץ על Bedrock AgentCore Runtime [15:38]. הסלייד עם הקוד מראה את ההרכב: ייבוא של Agent ו-tool מ-strands, BedrockModel לגישה ישירה ל-Bedrock ללא מפתחות API, BedrockAgentCoreApp כ-runtime, ולקוח Kubernetes.



כ-60 שורות לוגיקה: הקונפיגורציה של המודל, ה-SYSTEM_PROMPT ותיאור ארבעת שלבי ה-workflow שהסוכן אמור לבצע.

ה-system prompt קצר ומגדיר תפקיד אחד: סוכן אוטונומי לרמדיאציה של עלויות רשת ב-EKS, שמנטר תעבורה, מזהה אנומליות ומבצע פעולה לתיקונן. ה-workflow שמופיע בו בארבעה שלבים:

- קריאה ל-get_remediation_status כדי לבדוק את ה-thresholds
- אם תעבורת inter-AZ חצתה את הסף — קריאה ל-fix_inter_az_traffic
- אם תעבורת NAT Gateway חצתה את הסף — קריאה ל-create_vpc_endpoint
- דיווח על הפעולות שבוצעו, כולל החיסכון

הכלי הראשון, לפי הסבר רענן, מבצע Apply Traffic Distribution עם הערך PreferSameZone. הוא מציין שהקלאסטר רץ על Kubernetes 1.35 (מופיע גם במסך פרטי הקלאסטר בקונסול), גרסה שתומכת ביכולת הזאת. הכלי השני יוצר VPC Endpoint — במקרה שלהם Gateway Endpoint ל-DynamoDB.

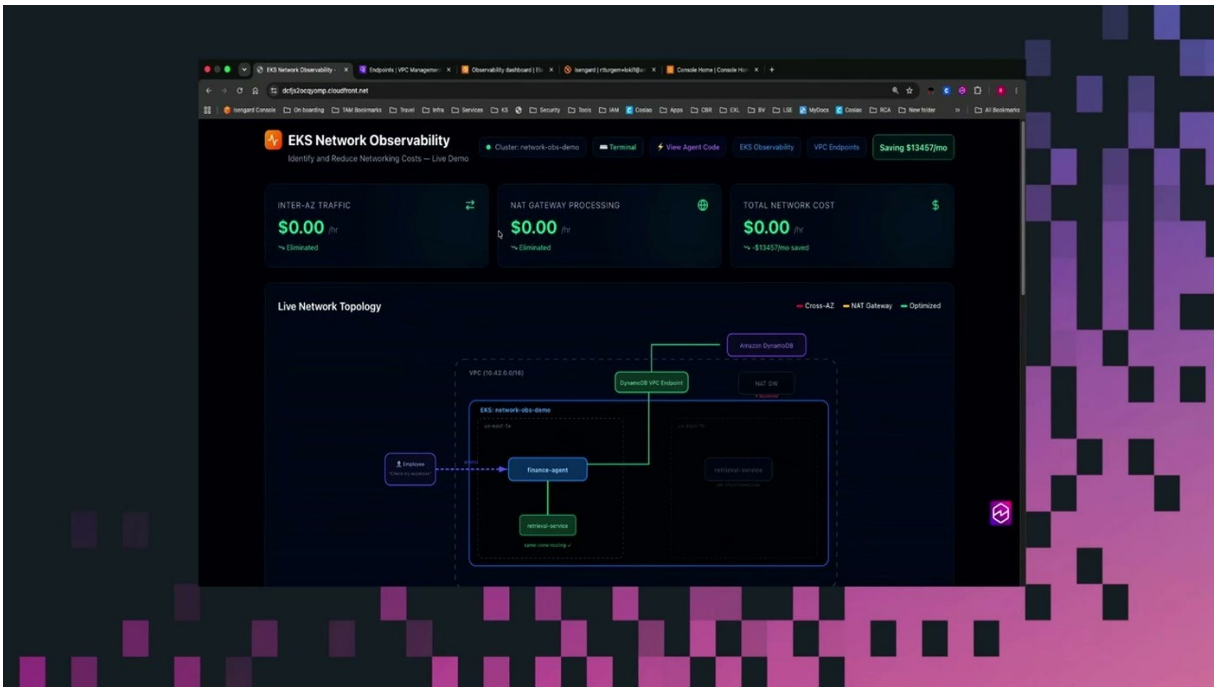
— רענן תורגמן [17:10] אנחנו נראה שבגדול מה שהוא יעשה זה Apply Traffic Distribution, Preferred Same Zone. אני רץ על EKS 135, מה שתומך בפריפר Same Zone

ההרצה

בהרצה החיה מוצגים Agent Trace ו-Agent Output זה לצד זה. הסוכן מזהה את ה-current Mbps של שני הצירים, קובע ששתי המטריקות חצו את הסף, ומפעיל את שני הכלים. ה-output מאורגן בארבעה שלבים — observe, reason, act, verify: האבחנה היא missing traffic distribution מצד אחד, ו-DynamoDB traffic through NAT gateway, no VPC endpoint מצד שני; אחריהן שתי הפעולות והוריפיקציה שלהן.

— רענן תורגמן [17:10] הוא מזהה ששתי המטריקות exceeded threshold והוא לוקח action. הוא מריץ fix inter-AZ traffic ו create VPC endpoint

9. התוצאה והאימות

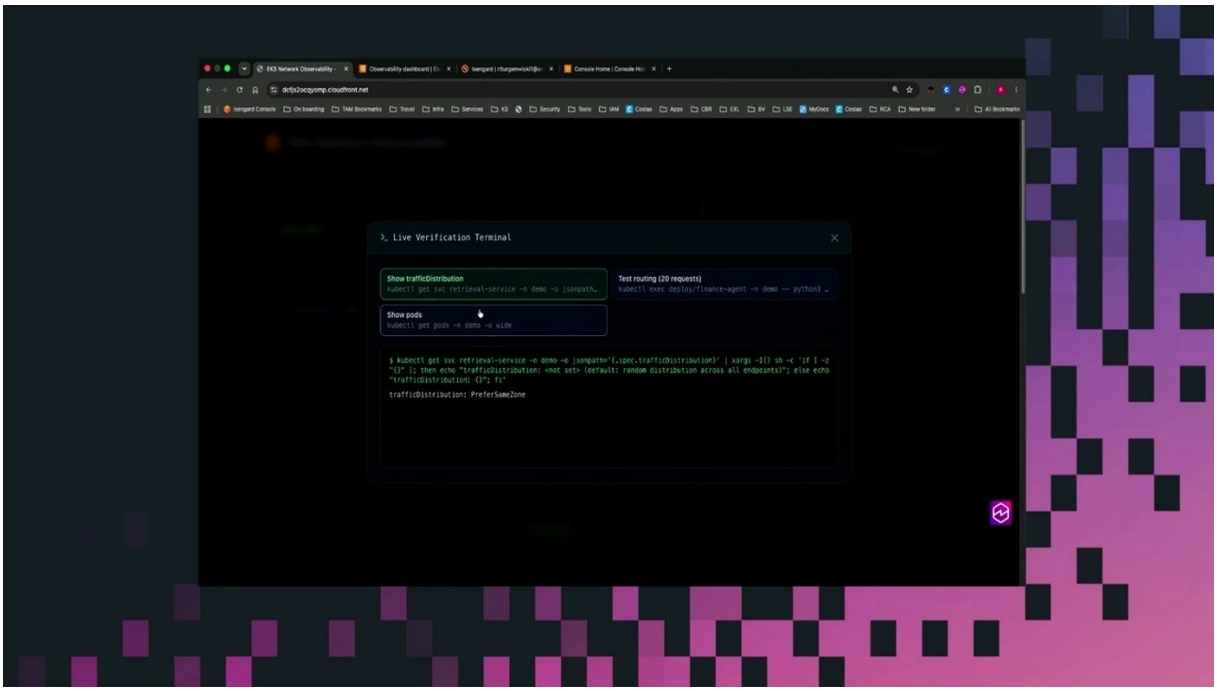


אחרי הרמדיאציה: שני מדדי העלות על $0.00 לשעה ומסומנים Eliminated, והטופולוגיה מראה את ה-DynamoDB VPC Endpoint ואת ה-retrieval service המרוחק מנוטרל.

הדשבורד מציג אפס בשני הצירים ותג חיסכון של $13,457 לחודש. בטופולוגיה רואים שהתעבורה ל-DynamoDB עוברת עכשיו דרך ה-VPC Endpoint ולא דרך ה-NAT GW, ושה-Finance Agent מדבר רק עם ה-retrieval service שבאותו AZ, עם הערת same-zone routing.

האימות נעשה בשלוש דרכים: הרצה חוזרת של 20 הבקשות, בדיקת ה-trafficDistribution ב-kubectl, ומבט בקונסול ה-VPC על ה-endpoint שנוצר.

— רענן תורגמן [18:47] הפעם כל ה-20 בקשות הגיעו ל-retrival service באותו availability zone

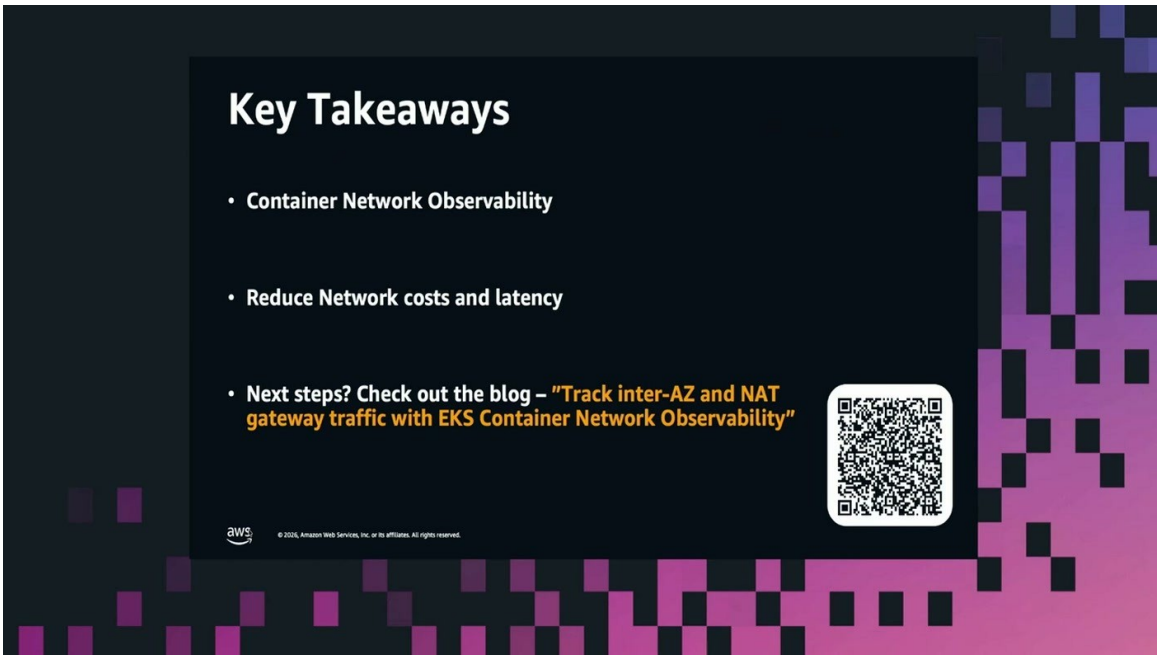


טרמינל האימות: הפקודה kubectl get svc retrieval-service עם jsonpath על spec.trafficDistribution, והפלט trafficDistribution: PreferSameZone.

בקונסול ה-VPC נראה ה-DynamoDB Endpoint שנוצר שלוש דקות קודם לכן — כלומר על ידי הסוכן, בזמן הסשן עצמו. רענן מסכם שהסוכן רץ במקביל למערכת כל הזמן, ומייצר רמדיאציה 24/7 בלי תלות במקום שבו ה-pods במקרה נחתו.

סייג רענן מוסיף הסתייגות חשובה: תעבורת inter-AZ אינה בהכרח תקלה. יש מקרים שבהם רוצים לפזר בין AZs מסיבות זמינות, ולכן ההחלטה של הסוכן נשענת על threshold שהמפעיל מגדיר — ולא על כלל גורף. "ואם יש לי אינטר-AZ traffic, מה שלא אומר שזה לא בסדר, כלומר יש מקרים שאנחנו רוצים להתירות ועוד אינטר-"AZ traffic [18:47].

10. מה לוקחים מכאן



שקופית הסיכום עם שלוש הנקודות והפניה לבלוג המלווה, כולל ברקוד לסריקה.

- **לבדוק אם הקלאסטרים שלנו כבר יכולים להפעיל את זה.** היכולת היא add-on בקונסול ולא פרויקט. עלות ההתנסות נמוכה: אין שינוי ב-deployments, אין sidecars, וה-agent צורך מעט משאבים. שווה להריץ על קלאסטר לא-פרודקשן ולראות מה הטבלה מראה.
- **לחפש קודם את ה-NAT Gateway, לא את ה-inter-AZ.** בדמו הפער היה כמעט פי חמישה לטובת ה-NAT Gateway ($15.53 מול $3.16 לשעה). תעבורה לשירותי AWS שיוצאת דרך NAT ולא דרך VPC Endpoint היא בדרך כלל הפירות הנמוכים.
- **trafficDistribution הוא הגדרה קיימת, לא מוצר.** PreferSameZone ברמת ה-Service הוא תיקון בן שורה אחת, ודורש גרסת Kubernetes תומכת (בדמו: 1.35). את זה אפשר ליישם גם בלי שום סוכן.
- **סוכן הרמדיאציה הוא הדגמה, לא מוצר מדף.** הוא נכתב לצורך הסשן ב-Strands Agents SDK על Bedrock AgentCore Runtime, וה-system prompt שלו כולל שורת "You are AUTHORIZED to make changes". בסביבה מפוקחת, הרשאה לסוכן לבצע שינויים בתשתית היא החלטת ממשל ולא החלטה טכנית.
- **המדידה היא הערך העיקרי; האוטומציה היא הבונוס.** גם בלי הסוכן, הידיעה מי מדבר עם מי ברמת pod ולאיזה AZ היא מה שהיה חסר. איתי: "ראיתם שהפודים שלכם יכולים לקבל ניטור מבלי הצורך לשנות את הדיפלוימנטים לא מוסיפים Side Cows, אין שכבות וירטואליזציה" [20:22].

בסיום מפנים הדוברים לבלוג מקיף של AWS בנושא, שכותרתו מופיעה על שקופית הסיכום: "Track inter-AZ and NAT gateway traffic with EKS Container Network Observability".

מילון מונחים

מונח	מה זה	איך הוא נשמע בתמלול
Container Network Observability	סט יכולות ניטור רשת מובנה ב-Amazon EKS	קונטיינר נטוורק אובזרוביליטי
Network Flow Monitor Agent	ה-DaemonSet שאוסף את ה-flows ברמת ה-node	נטוורק פלו מוניטורינג אג'נט
eBPF	מנגנון בקרנל שמאפשר איסוף נתונים ללא שינוי באפליקציה	IBPF
Flow	זרימת תעבורה בין מקור ליעד; נאספים 500 המובילים ל-node	פלור / פלורס
Service map	תצוגת גרף של הקשרים בין השירותים בקלאסטר	סרוויס מפ

מונח	מה זה	איך הוא נשמע בתמלול
Flow table	טבלת top talkers עם AZ מקומי ומרוחק וכמות דאטה	פלו טייבל
Historical Explorer	תצוגת CloudWatch להיסטוריית ה-flows עם חיתוכים נוספים	היסטוריקל אקספלורר
Inter-AZ traffic	תעבורה בין availability zones, מחויבת בשני הכיוונים	אינטר איזי טראפיק
NAT Gateway	יציאה לאינטרנט הציבורי; היקרה מבין דרכי התעבורה בדמו	נאט גטווי / נאד גטווי
VPC Endpoint	גישה פרטית לשירות AWS בלי מעבר באינטרנט	וי-פי-סי אינדפוינט
trafficDistribution: PreferSameZone	הגדרת Service ב-Kubernetes שמעדיפה endpoints באותו AZ	פריפרד סיים זון
EKS Pod Identity / IRSA	שתי דרכים להצמיד IAM Role ל-workload ב-EKS	איקס פוד איידנטיטי / אירסה
Strands Agents SDK	ה-SDK שבו נכתב סוכן הרמדיאציה	סטרנד SDK
Bedrock AgentCore Runtime	ה-runtime המנוהל שעליו רץ הסוכן	אג'נט קור ראנטיים