מכיבוי שריפות לתפעול פרואקטיבי עם AWS DevOps Agent

AWS Summit Tel Aviv 2026 ,סיכום סשן — TLV-SEC302

Ruthy Ahuvi, Head of Enterprise Support, Israel & Sub-Saharan Africa, AWS

Noam Lichtblau, Principal Technical Account Manager, AWS | Aviv Karbol, DevOps Architect, Infra & AI Platform, Skai

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

על המסמך הזה

המסמך מסכם את הסשן TLV-SEC302 מתוך AWS Summit Tel Aviv 2026, מסלול AWS AI Cloud Ops, Infrastructure & Security. הוא נבנה מתמלול מלא של ההקלטה ומהסליידים שחולצו מהווידאו, כך שאפשר לחזור לתוכן בלי לצפות שוב בארבעים ואחת הדקות. חותמות הזמן בסוגריים מפנות לנקודה המדויקת בהקלטה.

הסשן מורכב משלושה חלקים: הצגת השירות (Ruthy Ahuvi), דמו חי מקצה לקצה של חקירת תקלה (Noam Lichtblau), וסיפור לקוח מהשטח של חברת Skai (Aviv Karbol).

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה הוצג, ומה באמת חדש כאן.
- **פרקים 2–4 — מה השירות ואיך הוא בנוי:** ארבעת ה"כובעים", ארכיטקטורת הסוכן, ושכבות הקונטקסט.
- **פרקים 5–7 — הדמו:** הזרקת באג לפרודקשן, החקירה האוטונומית, ותוכנית המיטיגציה.
- **פרקים 8–10 — מספרים, מניעה פרואקטיבית ופיצ'רים חדשים:** מה המוצר מספק מעבר לתגובה לאירועים.
- **פרקים 11–12 — Skai ותמחור:** לקוח ישראלי שמריץ את זה בפרודקשן, ומודל העלות.
- **פרק 13 — מה לוקחים מכאן:** מסקנות ונקודות להמשך, ומילון מונחים בסוף.

> **הערת מקור** המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך מונחים לועזיים ושמות מוצרים הופיעו בו בשיבוש פונטי ותוקנו ידנית מול הסליידים. הציטוטים נבדקו מול ההקלטה בחותמת הזמן המצוינת ומופיעים כפי שנאמרו. מספרים שמופיעים על הסליידים צוינו ככאלה.

שקף הפתיחה: שלושת הדוברים ותפקידיהם — שניים מ-AWS ואחד מצד הלקוח.

1. תקציר מנהלים

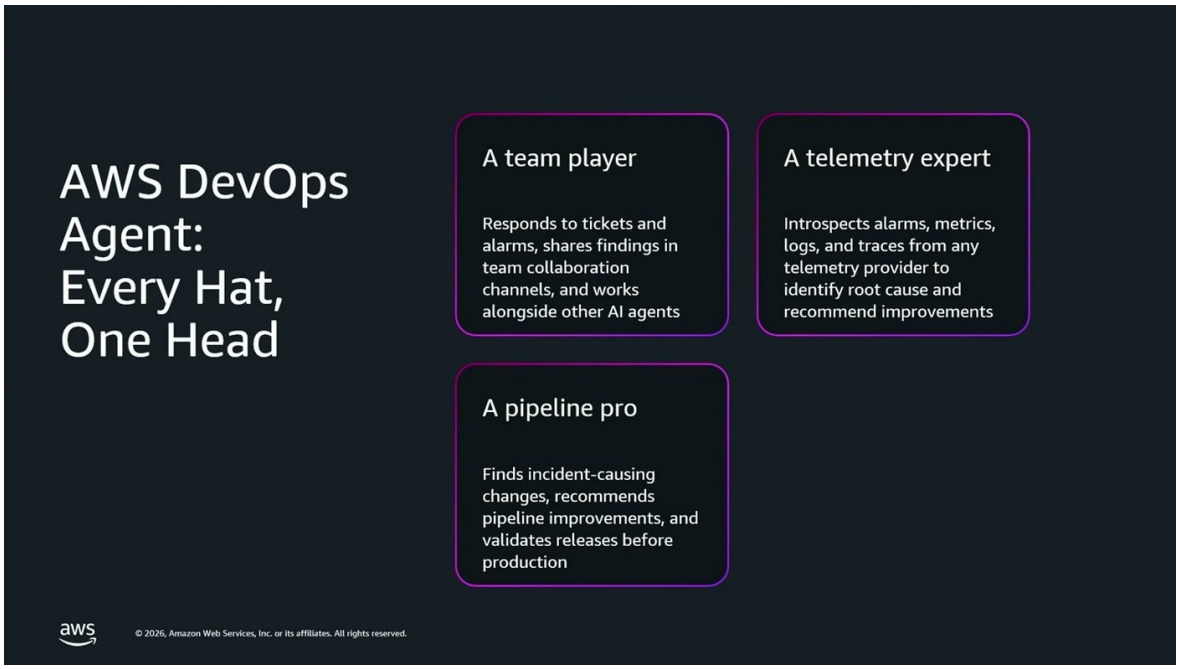
הסשן מציג את AWS DevOps Agent — שירות שהוכרז ב-re:Invent האחרון — כ"מהנדס SRE" אוטונומי שמצטרף לצוות התפעול. הטענה המרכזית אינה שהסוכן חכם יותר מאדם, אלא שהוא זמין 24/7, מחזיק בו-זמנית את הטופולוגיה, הטלמטריה והקוד, ולכן מגיע ל-root cause מהר בסדר גודל. ההרצאה לא נשארה ברמת ההבטחה: היא כללה דמו חי שבו הוזרק באג לפרודקשן, וסיפור לקוח עם מספרים.

- **הנימוק לקיום המוצר הוא קצב הפיתוח החדש, לא התקלות עצמן.** התזה: כשצוותי פיתוח מייצרים קוד פי עשר מהר יותר בעזרת AI, מספר הדיפלוימנטים והשינויים גדל, ואיתו נקודות הכשל. "בסשן היום נדבר איך אתם יכולים לעבור מקיבוי שריפות ריאקטיבי למצוינות פרואקטיבית" [1:32].
- **הסוכן בנוי סביב ההנחה שכל אירוע מקורו בשינוי.** "אנחנו בסוף מבינים שכל אירוע נובע מאיזשהו שינוי, אם זה שינוי בקוד האפליקציה או שינוי בקוד התשתית" [4:34] — ולכן יש לו אינטגרציות מובנות ל-GitHub, GitLab ו-Azure DevOps, לצד כלי ניטור.
- **בדמו הוא פיצח באג אמיתי בשש דקות.** הוזרק שדה timestamp לרשומת cache, שהשבר את הדה-סריאליזציה אצל הצרכנים. הסוכן זיהה את הקומיט, את שורות הקוד ואת קצב השגיאות (כ-60 שגיאות בדקה) — 5:51 דקות מרגע ההפעלה ועד ה-root cause.
- **המספרים שהוצגו מבוססים על 20+ לקוחות ושותפים.** קיצור של 75% ב-MTTR, 80% קיצור בזמן החקירה, ו-94% דיוק בזיהוי ה-root cause (מהשקף; טווחים: 50–92%, 84–100%) [21:37, 23:08].
- **לקוח ישראלי כבר מריץ את זה בפרודקשן.** Skai בנתה מוצר חדש ב-90 יום עם איש DevOps אחד + הסוכן. "אז דיבופס אחד, אני, ודיבופס אייג'נט" [33:58]. תקלה שהצוות רדף אחריה 16 שעות נסגרה ב-22 דקות אחרי שהסוכן נכנס.
- **התמחור הוא pay-per-use על זמן עבודה בפועל.** "אתם משלמים על כל שנייה שבה הסוכן מבצע חקירה, על כל שנייה שאתם בצ'ט מולו. כשהוא לא עובד, אתם לא משלמים עליו" [38:44], בתוספת free trial וקרדיטים ללקוחות תוכניות התמיכה.

מה חדש כאן באמת

כלי ניטור כבר יודעים לזהות אנומליות — זה לא החידוש. החידוש הוא שכלי הניטור (בדמו: Dynatrace) מפעיל את הסוכן כדי שיחקור, והסוכן מחזיר לא התראה אלא דוח תחקיר עם root cause ותוכנית מיטיגציה בת ארבעה שלבים. כפי שנוסח בהרצאה: "אתם כבר לא מתעסקים כאן אך ורק בלקבל התראות ממערכות, אתם מקבלים כאן סיכום חקירה מוכן על מה בדיוק קרה ואיך לפעול על פיו" [13:54].

2. מה זה AWS DevOps Agent — ארבעה כובעים



ארבע התכונות שסביבן נבנה השירות, כפי שהוצגו על השקף: שחקן צוות, מומחה טלמטריה, ומומחה pipeline.

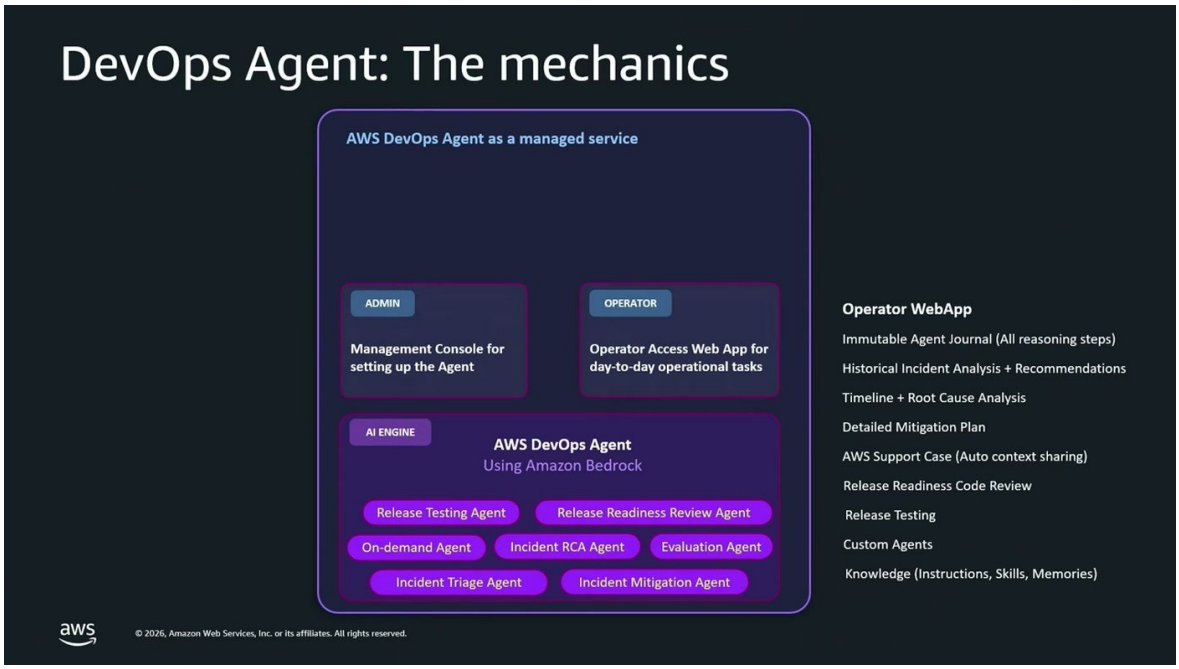
השירות הוגדר כ-Frontier Agent שפועל בצורה אוטונומית: "הוא חבר הצוות הטיפולי החדש שלכם שזמין 24 שבע" [3:03]. הוא פותר ומגיב לאירועים, מייעל אמינות וביצועים, מטפל במשימות SRE לפי דרישה, וביוני האחרון נוספה לו יכולת Release Management. ארבעת ה"כובעים" שהוצגו:

- **A team player — חבר צוות.** מגיב לטיקטים, אלרטים ו-pages ממערכות חיצוניות כמו ServiceNow ו-PagerDuty, ומתקשר ממצאים בטיקט המקורי או בערוצי Slack / Microsoft Teams. הוא גם יכול להתחבר לסוכני AI אחרים, למשל Dynatrace Davis.
- **A telemetry expert — מומחה טלמטריה.** אינטגרציות built-in לא רק ל-Amazon CloudWatch אלא גם ל-New Relic, Grafana, Datadog, Dynatrace ו-Splunk, וכן לסטאק הפתוח Prometheus + Loki או כל מערכת אחרת, דרך MCP Server מובנה [3:03, 4:34].
- **A pipeline pro — מומחה שינויים.** אינטגרציות ל-GitHub, GitLab ו-Azure DevOps כדי להבין את תהליכי הדיפלוימנט, לזהות מה מקור השינוי שגרם לאירוע, ולהנחות כיצד לבצע rollback. במקביל הוא ממליץ אילו בדיקות ואישורים להוסיף ל-pipeline כדי למנוע אירועים עתידיים [4:34].
- **Knows your apps — מכיר את האפליקציה.** "הרי it's all about the context" [4:34]. הסוכן מייצר ומתחזק אוטומטית application topology — גרף של הישויות בסביבה וההקשרים ביניהן.

מסלול מילוט לאדם

נקודה שקל לפספס: בכל שלב — בזמן חקירה, בבדיקת ממצאים או בבניית תוכנית מיטיגציה — אפשר בלחיצת כפתור לפנות למהנדס מארגון התמיכה של AWS, והוא מקבל את כל הקונטקסט של החקירה שבוצעה עד אותה נקודה [6:05]. זו נקודת חיבור ישירה לתוכניות ה-Support הקיימות, ולא ערוץ נפרד.

3. איך זה בנוי מאחורי הקלעים



המבנה כשירות מנוהל: שכבת האינטגרציות, מנוע הסוכן על Amazon Bedrock עם תתי-הסוכנים, ושני ממשקי הגישה.

Agent Space והטופולוגיה

נקודת ההתחלה היא מרחב עבודה: "ברגע שאתם מפעילים DevOps Agent ומגדירים לו מרחב עבודה בשם Agent Space, בתוכו אתם נותנים לו גישה לחשבון ענן אחד או יותר בעזרת IAM Role" [6:05]. משם הוא מתחיל לסרוק, לגלות רכיבים ולמפות את ההקשרים ביניהם.

המיפוי הוא תהליך מתמשך ולא סריקה חד-פעמית. אם מחברים מקור טלמטריה, הוא ממפה log groups ומטריקות לישויות בטופולוגיה; אם מחברים CI/CD pipeline, הוא ממפה דיפלוימנטים לישויות [6:05]. התוצאה המעשית: הוא מצמצם את מרחב השאילתות שעליו להריץ כדי להגיע למקור הבעיה. במקביל הוא משווה את המפה לתפיסות קיימות — Best Practices של AWS ועקרונות Well-Architected — ומייצר המלצות שיפור [7:39].

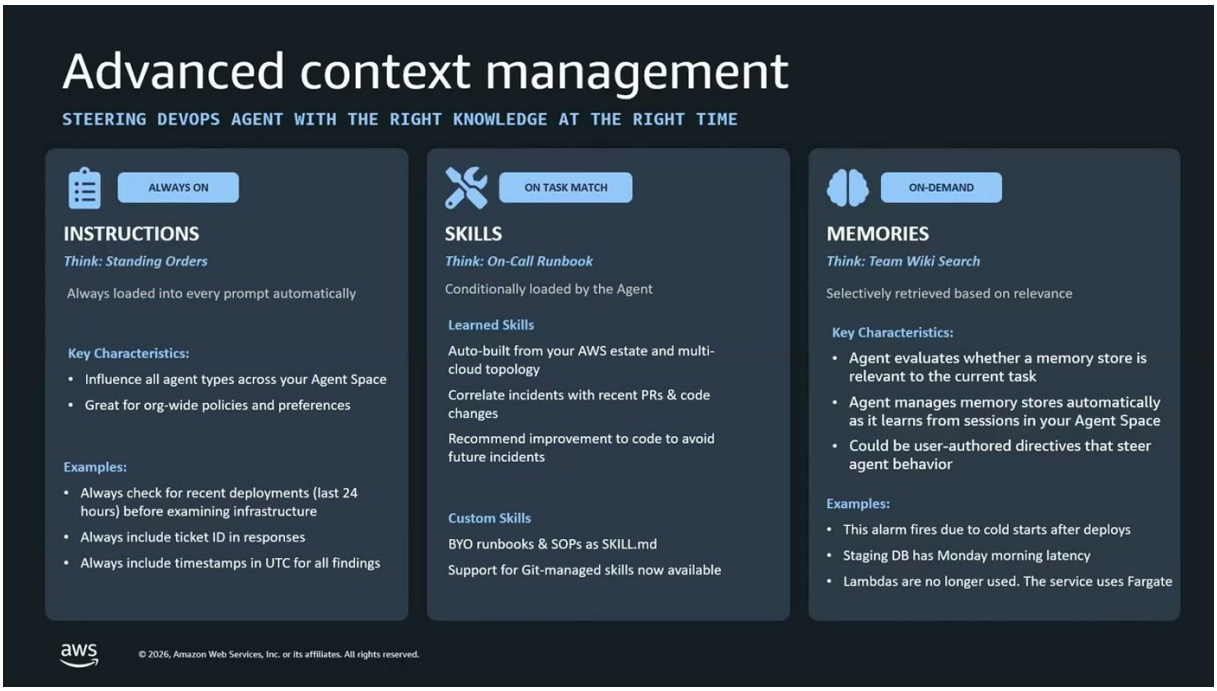
אורקסטרטור של תתי-סוכנים

בליבה יושב מנוע ה-AI של DevOps Agent, הבנוי על גבי Amazon Bedrock. המבנה הוא אורקסטרטור שמורכב ממספר sub-agents, כשכל אחד אחראי על תהליך מסוים: אחד ל-triage ראשוני, אחד לבניית RCA, אחד לתוכניות מיטיגציה [7:39]. בשקף מופיעים בין היתר Incident Triage Agent, Incident RCA Agent, Incident Mitigation Agent, Evaluation Agent, On-Demand Agent, Release Testing Agent ו-Release Readiness Review Agent. ההדגשה הייתה שאין צורך להכיר או לנהל אותם — המערכת מתזמרת אותם אוטומטית.

שני ממשקים, ומצב headless

- **AWS Management Console — שכבת הניהול.** כאן מגדירים את ה-Agent Space ואת האינטגרציות: webhooks ממערכות טיקטים, כלי CI/CD, observability, וערוצי תקשורת [7:39].
- **Operator Web App — העבודה היומיומית.** הצוותים מנהלים כאן את החקירה, מתקשרים עם הסוכן, מקבלים המלצות RCA ורואים את הטופולוגיה [9:16].
- **Remote server — גישה תוכניתית.** הסוכן נחשף דרך פרוטוקולי API, A2A ו-MCP, כך שכל coding agent — Kiro, Claude Code או כל agentic harness אחר — יכול לקרוא לו לסקירת מוכנות לשחרור, ניהול תקלה או שאלות על בריאות המערכת, בלי קוד אינטגרציה מותאם [9:16].
- **אוטומציה והרשאות.** חקירות ומיטיגציות זמינות גם דרך Amazon EventBridge; הגישה עוברת דרך AWS IAM Identity Center וספקי זהות חיצוניים [9:16].

4. ניהול קונטקסט: שלוש שכבות ידע



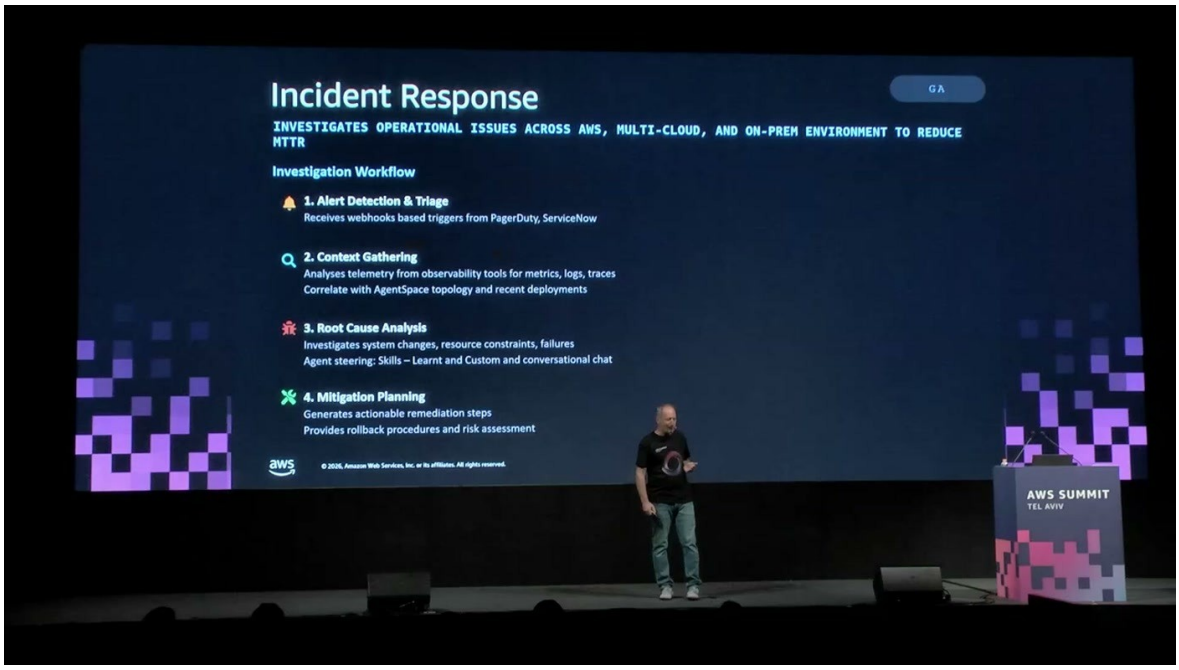
שלוש שכבות הידע וההבדל ביניהן — מתי כל אחת נטענת ומה מתאים לשים בה.

זה החלק הרלוונטי ביותר למי שמתכנן להתאים את הסוכן לארגון שלו. הוצגו שלוש שכבות, שכל אחת מתנהגת אחרת מבחינת מתי היא נטענת:

- **Instructions — always on.** נשמרות כקובצי AGENT.md ומוזרקות לכל אינטראקציה, בלי שהסוכן מפעיל אותן. מתאימות למדיניות ארגונית רוחבית. דוגמאות מהשקף: תמיד לבדוק דיפלוימנטים מ-24 השעות האחרונות, תמיד לכלול ticket ID בתשובה, תמיד להציג חותמות זמן ב-UTC [10:51].
- **Skills — on task match.** "זה מה שהופך את הסוכן שלכם ממומחה כללי למומחה בתשתית שלכם" [10:51]. שני סוגים: Learned Skills שהסוכן בונה בעצמו מסריקת החשבונות, הריפוזיטוריז, ה-pipelines והטלמטריה — "אפס עבודה מצדכם"; ו-Custom Skills, הרנבוקים וה-SOP שלכם, שנשמרים כקובצי SKILL.md (כולל תמיכה ב-skills מנוהלי Git, לפי השקף).
- **Memories — on demand.** עובדות שנשלפות לפי רלוונטיות, בניגוד ל-Skills שהן ידע פרוצדורלי. Managed Memories שהסוכן מייצר לבדו (למשל: לשמור את ה-root cause של כל אלרם, כדי שבפעם הבאה יגיע מהר לסיבות הסבירות), ו-User Authored — מה שאתם אומרים לו בצ'אט "תזכור ש..." [12:22].

> **Ruthy Ahuvi, [12:22] —** הכלל הוא פשוט, אם זה כל אינטראקציה שימו אותם ב-instructions, אם זה how to למשימה מסוימת שימו אותה ב-skills, ואם זה עובדה שימושית שרלוונטית שימו אותה ב-memories

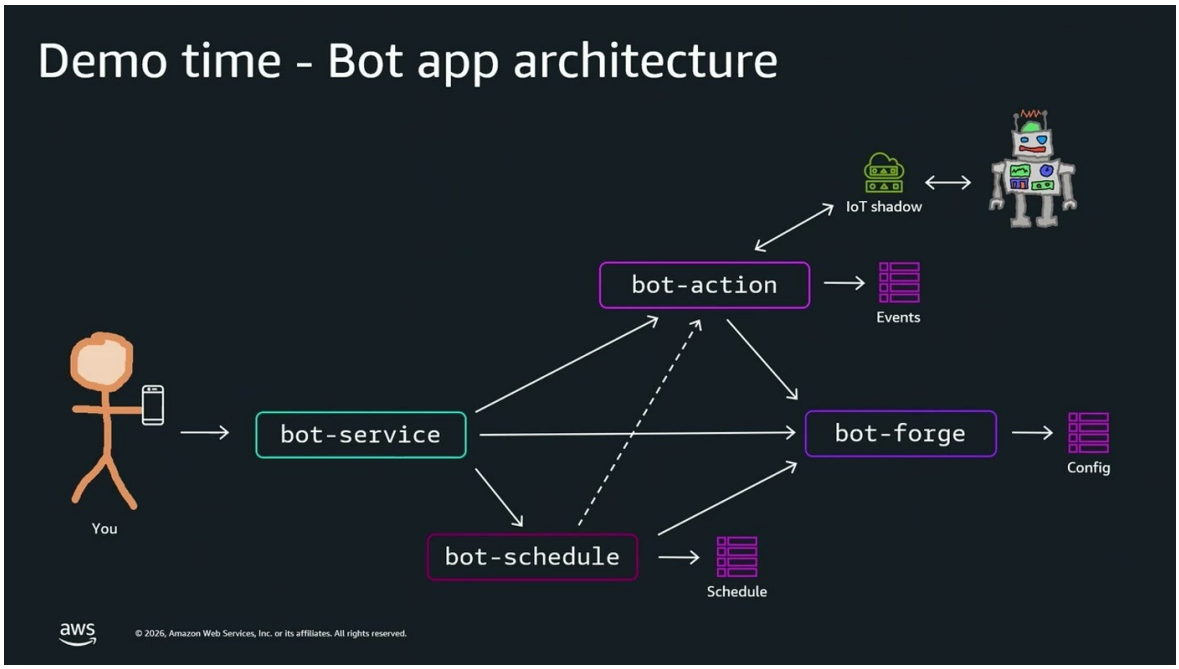
5. Incident Response — ארבעת שלבי החקירה



מבנה החקירה כפי שהוצג לפני הדמו: מהטריגר, דרך איסוף וקורלציה, ועד ניתוח שורש ותוכנית מיטיגציה.

- **1. Alert Detection & Triage — התנעה.** webhook מכלי הניטור או מכלי הטיקטינג, או חקירה אד-הוק שיוזמים מתוך הצ'אט: "תגידו ל-DevOps Agent, אני רוצה שתבצע לי חקירה על בעיית הביצועים שהייתה ביום שני בערב, בה מערכת עבדה לאט, והוא יתחיל חקירה" [13:54].
- **2. Context Gathering — איסוף.** לוגים, מטריקות, traces ,trails ושינויים בקוד — לפי מה שחיברתם אליו.
- **3. Root Cause Analysis — קורלציה והצלבה.** "הוא גם מסביר לנו סטפ אחרי סטפ בדיוק מה הוא עושה, מה הוא הולך לבדוק, ממש כמו מהנדס SRE שבונה כאן דוח תחקור" [13:54].
- **4. Mitigation Planning — תוכנית פתרון.** מקור הבעיה יכול להיות inputs, באגים בתוכנה, או APIs ששרשרו — והתוכנית נבנית בהתאם.

6. הדמו: מזריקים באג לפרודקשן



אפליקציית ההדגמה: ארבעה web services — bot-service כ-frontend, bot-action ו-bot-schedule, ו-bot-forge שמקצה רובוטים למשימות.

הדמו רץ על מערכת פשוטה יחסית שמקצה רובוטים למשימות. מאחורי ארבעת השירותים יש דאטאבייסים, autoscalers, load balancers, security groups וקוד — "וכל אחד מהם הוא מוקד פוטנציאלי לתקלה" [15:27].

התרחיש: בתוך bot-forge יש מטמון (cache) לקונפיגורציית הרובוטים. הדובר הוסיף לרשומת ה-cache שדה timestamp, כביכול כדי לראות כמה זמן רשומות נשארות שם — תוך התעלמות מודעת מכך שיש צרכנים לאותו cache שלא מצפים לשדה חדש. משם: commit ל-GitHub, GitHub Action, ו-CI/CD pipeline דוחף לפרודקשן.

> **Noam Lichtblau, [15:27] —** אבל אני אופטימיסט... אני מקווה לטוב, נכון? מה כבר יכול להישבר?

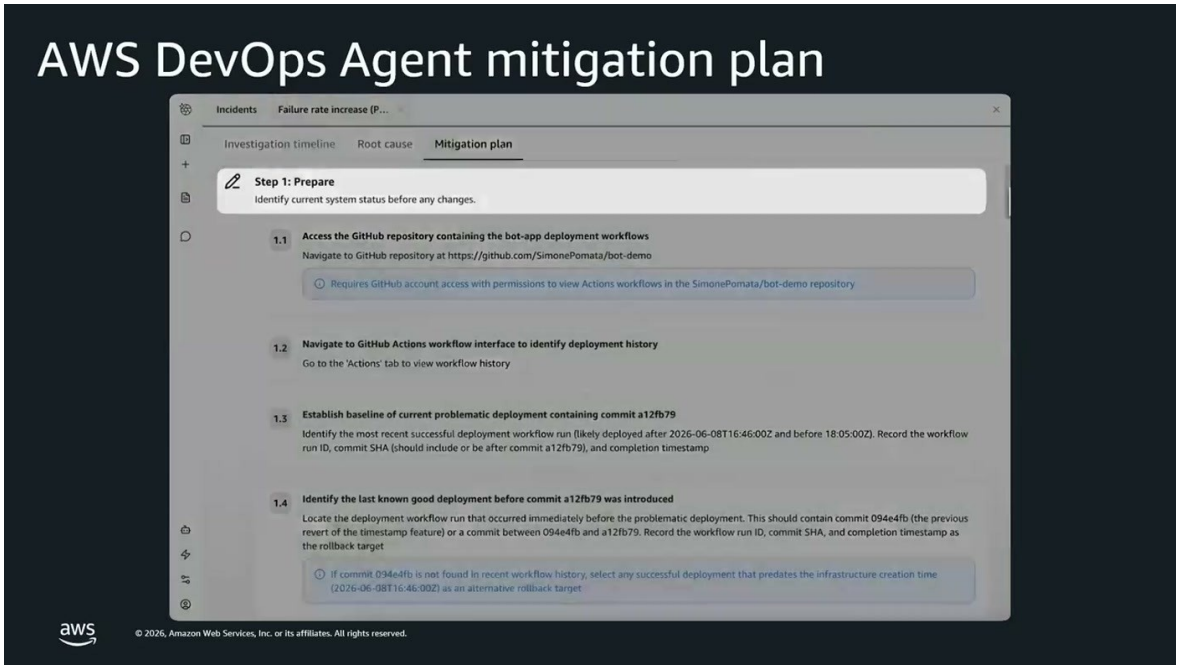
מה שנשבר: Dynatrace זיהה כמות חריגה של שגיאות בשלושה web services מול ה-thresholds שהוגדרו. עד כאן — כלי ניטור עושים את זה שנים. החידוש: "מה שכן חדש כאן עכשיו זה ש-Dynatrace הולך להשתמש ב-DevOps Agent כדי לבקש ממנו לתחקר את מקור הבעיה הזאת" [17:01]. המבנה שהועבר לסוכן היה סטנדרטי — שלושה שירותים, כמות השגיאות חרגה מה-threshold, זה הקונטקסט להתחיל ממנו.

7. מה הסוכן מצא, ואיך

הדובר הציג קודם את התוצאה ורק אחר כך את הדרך. ה-root cause שהוחזר: בקומיט ספציפי התווסף שדה timestamp לרשומות ה-cache; השדה לא עובר דה-סריאליזציה אצל הצרכנים; נוצרו מאות שגיאות בקצב של כ-60 שגיאות בדקה — כולל ציון שורות הקוד שבהן זה קרה. הזמן מרגע הבקשה ועד ה-root cause: 5 דקות ו-51 שניות [17:01, 18:31].

מסלול החקירה בפועל

- **קודם הקשר, אחר כך מצב ריצה.** הסוכן ניגש ל-Agent Space וקרא את טופולוגיית השירות, ואז הפעיל APIs של AWS — EC2 DescribeInstances, DescribeLogGroups — כדי להבין את המצב בפועל ואיפה הלוגים [18:31].
- **ארבע בדיקות במקביל.** קריאת לוגים של שני רכיבים, מטריקות שימוש ב-load balancer, וסריקת ה-code repositories שנפתחה אליהם גישה [18:31].
- **כ-4 דקות לתוך החקירה — הממצא הראשון.** 50% מהשגיאות הן אותה שגיאה: ValueError שמופיע בלוגים של כל הרכיבים, כשרכיבים שמקבלים את רשומות ה-JSON מה-cache נכשלים ב-strict validation על שדה בשם timestamp [18:31].
- **חתך זמן.** השגיאות התחילו בשעה מסוימת; פניות ראשונות לרשומה, לפני cache hit, מצליחות — ואחרי cache hit מתקבלות השגיאות [20:05].
- **מעקב אחורה בשרשרת.** error rate גבוה של 500 ב-frontend, אבל הוא מזהה שהשגיאות מתפשטות (propagate) מרכיבים אחרים; הוא יורד לרכיבים, מזהה 500 ו-502, ומגיע ל-bot-forge — שם בוצע שינוי קוד [20:05].

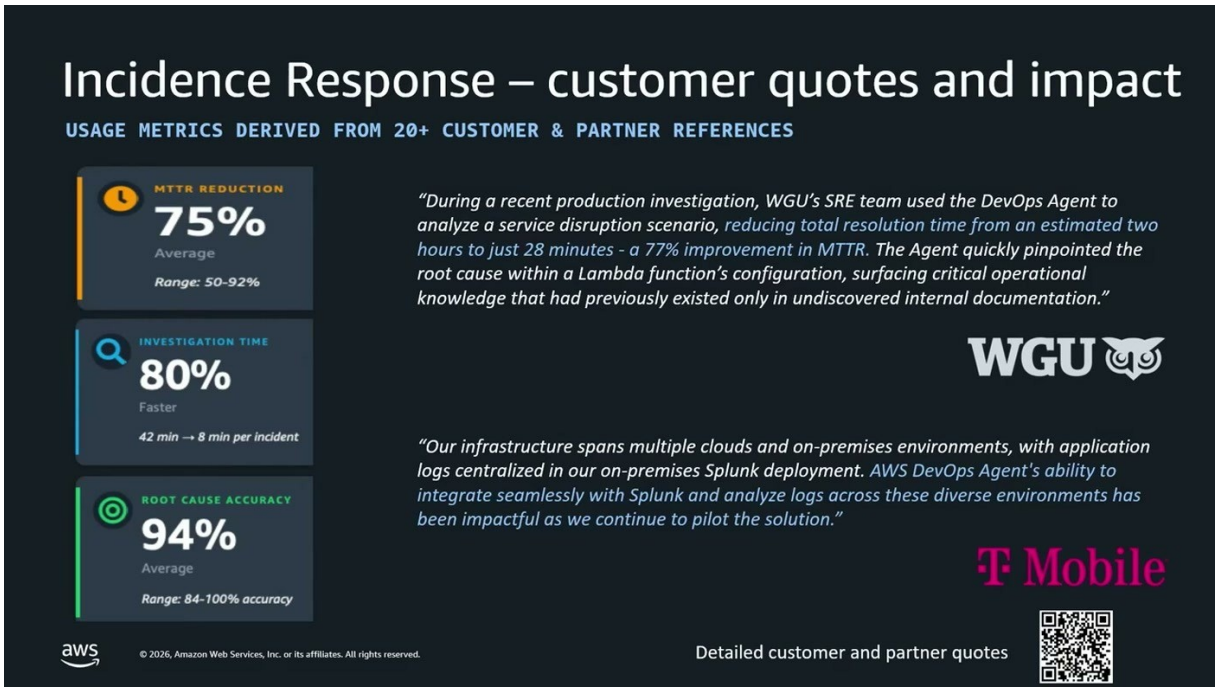


תוכנית המיטיגציה כפי שהוצגה בממשק — שלב ההכנה פתוח, עם הנחיות קונקרטיות לאיתור הקומיט.

המיטיגציה לא הייתה משפט אחד אלא תוכנית בת ארבעה שלבים [20:05, 21:37]: Preparation — לגשת ל-Git, למצוא את הקומיט הבעייתי ואת הקומיט שלפניו; Pre-validation — לוודא שהסביבה ערוכה לקלוט את השינוי, שהתקלה עדיין מתרחשת, ושמכונות ה-EC2 באוויר; Apply — הדרכה step by step, כולל Re-run all jobs ב-GitHub Actions כדי לדחוף את הקומיט הקודם חזרה לפרודקשן והמתנה לבנייה מחדש דרך ה-autoscaler; Post-validation — לאמת שהבעיה נפטרה. בפועל: refresh ב-Dynatrace הראה ירידה בכמות השגיאות מתחת לרף הקריטי, והאירוע נסגר.

> **Noam Lichtblau, [21:37] —** לא חקרתי בעצמי כלום, לא נכנסתי ללוגים, לא ניסיתי לייצר קורלציות. הוא עשה את העבודה במקומי וחסך לי הרבה זמן

8. המספרים מהשטח



שלושת המדדים שהוצגו, כולל טווחים וציטוטי לקוחות — WGU ו-T-Mobile.

המספרים הוצגו כנגזרים מסקר מול 20+ לקוחות ושותפים שכבר משתמשים במוצר:

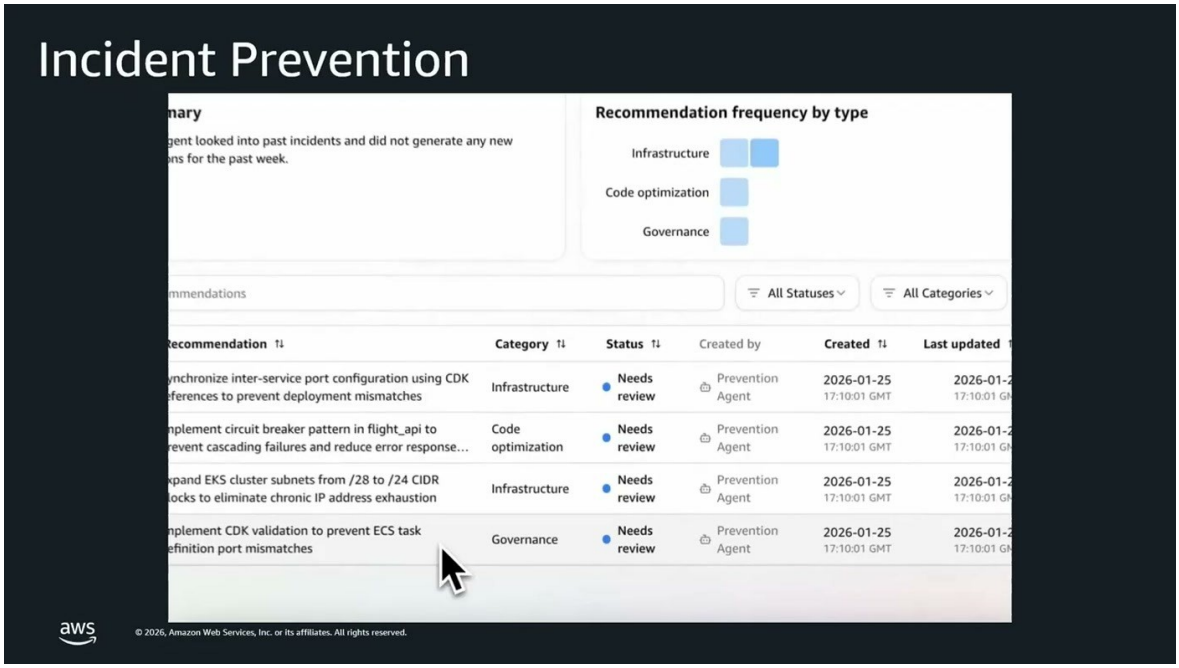
מדד	שיפור	טווח (מהשקף)
MTTR — זמן עד פתרון	קיצור של 75% בממוצע	92%–50%
זמן חקירה	מהיר ב-80%	42 דקות ← 8 דקות לאירוע
דיוק בזיהוי root cause	94% מהמקרים	100%–84%

Noam Lichtblau, [23:08] — הוא מקצר את זמן החקירה ב-80% בממוצע, והנתון הכי מרשים, ב-94% מהמקרים, הוא אכן מוצא את הרוט קוז הנכון לבעיות

פרשנות שימו לב להבחנה בין שני המספרים: 80% הוא קיצור זמן החקירה, 75% הוא קיצור ה-MTTR הכולל. ההפרש מייצג את מה שהסוכן עדיין לא עושה — את הביצוע בפועל של המיטיגציה, שבדמו נותר בידי האדם.

9. מניעה פרואקטיבית, לא רק תגובה

אחת לשבוע הסוכן סוקר את כל החקירות שביצע ואת הפעילות במערכת, ובונה המלצות שיפור — ואפשר גם לבקש זאת אד-הוק דרך הצ'אט [23:08]. ההמלצות מגיעות מארבעה תחומים: שיפורי monitoring וטלמטריה (הוספת מטריקות שאינן מנוטרות, שמירת לוגים לפרקי זמן ארוכים יותר); שינויים ברמת הסביבה (קונפיגורציה של רכיבים, load balancers, right-sizing של מכונות); שיפורי deployment pipeline (למשל אזורים שבהם לא מתבצעות בדיקות); ו-application resilience — למשל הוספת exponential backoff מול throttling מ-API של צד שלישי [23:08, 24:41].



מסך ההמלצות: backlog מדורג לפי קטגוריה וסטטוס, עם תדירות ההמלצות לפי סוג.

הדוגמה שהוצגה: port mismatch ב-ECS

הסוכן זיהה ש-ECS Tasks שנוצרים דרך CDK נפתחים עם port 8081 ל-health check בזמן ש-ECS משתמש ב-8080. התוצאה: ECS לא זיהה שה-tasks עלו, והדליק אותם שוב ושוב בלולאה. ההמלצה הייתה להוסיף ל-CDK זיהוי דינמי של הפורטים [24:41].

- **Generate agent spec — מהמלצה לקוד.** "בלחיצת כפתור אני אקבל כאן פרומט שאני יכול לקחת לתוך כלי הפיתוח שלי" [24:41] — Kiro, Claude, Cursor או אחר — כך שאפילו את שינוי הקוד לא צריך לבצע ידנית.
- **הצדקה לאחור.** לצד ההמלצה מוצגים האירועים שקרו בסביבה ושכנראה היו נמנעים אילו ה-guardrail הזה היה מיושם [24:41].
- **Backlog עם שני מסלולי סגירה.** לוחצים Implemented וההמלצה יורדת מה-backlog; לחלופין דוחים אותה ומסבירים מדוע — שיקולי עלות, מדיניות ארגונית — וההמלצות הבאות מותאמות לשיטות העבודה של הארגון [24:41, 26:15].
- **ה-skills גלויים לבדיקה.** דרך ה-Agent Space אפשר לפתוח את קובץ ה-Markdown ולראות מה נאסף ונלמד: טופולוגיה וארכיטקטורה, קונטיינרים שרצים, ה-critical path של רכיבים, ה-code repositories — ואפשר להוסיף לו מידע ידנית [26:15].

10. מה נוסף מאז ההכרזה

> **Noam Lichtblau, [26:15] —** אני לא ראיתי שבוע מאז ההכרזה שלא יצא פיצ'ר חדש ל-DevOps Agent

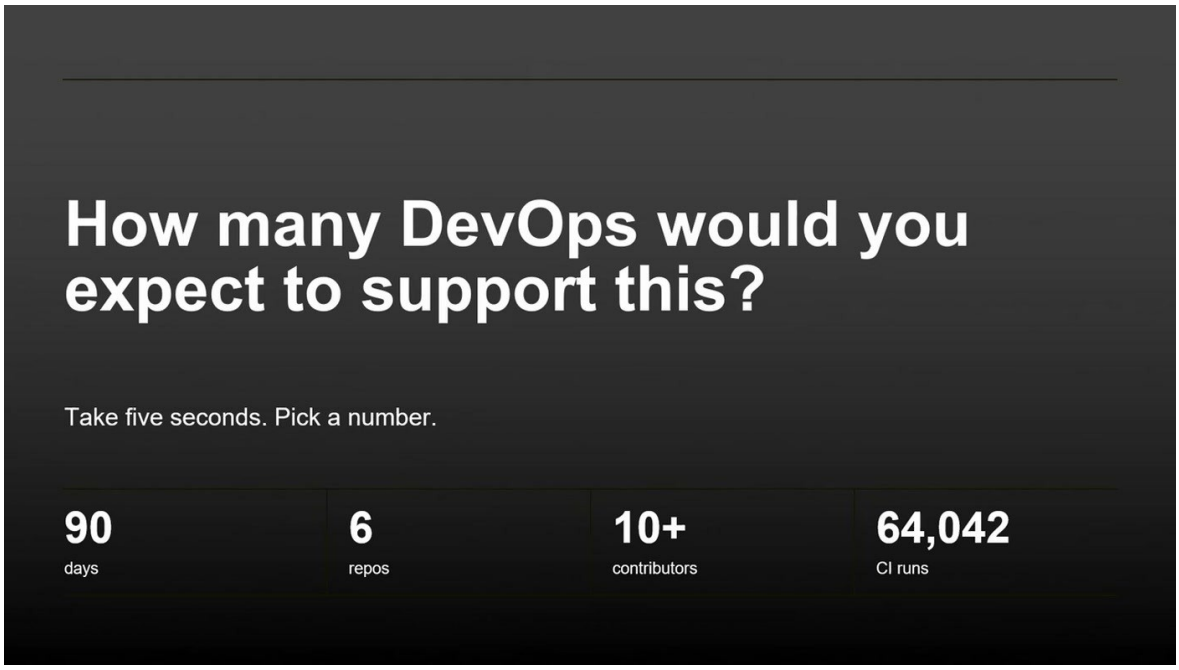
- **Triage Agent (מרץ).** פותר את מצב ה"תקלה אחת מייצרת חמש": הסוכן מכיר את הארכיטקטורה, מקשר בין תקלות ומסמן אותן כ-Linked. "היכולת הזאת מאפשרת לנו להתמקד בתקלה האמיתית, לפתור אותה, ואז התקלות משנה שלה שקשורות אליה גם ייפטרו" [27:47]. השקף מפרט גם את הצד הכלכלי — פחות חקירות כפולות ופחות בזבוז זמן.
- **צ'אט כמומחה DevOps, לא כ-Q&A על חקירות.** אפשר לשאול אותו על security, cost optimization או performance: אילו פונקציות Lambda עדיין רצות על Python 3.8, אילו certificates יפוגו בעוד שבועיים, מהן הזדמנויות ה-cost optimization המשמעותיות ביותר בדאטאבייסים בארגון [27:47].
- **Release Management (יוני, New York Summit).** שלושה מרכיבים: Release Readiness Review — הבנה בזמן כתיבת הקוד מה תהיה ההשפעה על הפרודקשן כפי שהוא רץ עכשיו, ולא כפי שהוא מצויר בתרשים ארכיטקטורה (dependencies, חריגה ממדיניות ארגונית, access control); Release Testing אוטונומי — הסוכן מזהה את השינויים, בונה בדיקות ומריץ אותן בסביבה ייעודית, מעין exploratory testing; ו-Shift Left מלא — הבאת הסוכן ל-IDE, ל-CI/CD pipelines ולכל pull request [27:47, 29:19].
- **Headless mode.** כדי שה-shift left יקרה, הסוכן הפך ל-MCP דו-כיווני ותומך ב-Agent-to-Agent Protocol, כך שניתן לחבר אותו לכל כלי שפתוח לפרוטוקולים האלה — ולבטל את ה-context switching של המפתחים [30:53].

יכולות ה-Release Management שהוכרזו ביוני, בסטטוס preview: readiness review, testing ו-shift left.

הנימוק שניתן ליכולות האלה היה כלכלי-הנדסי ולא טכנולוגי: "בעולם שבו קוד נכתב על ידי AI בקצב ובכמויות שבני אדם לא יכולים להספיק לבדוק" [27:47] — שיטות העבודה הקיימות פשוט לא מדביקות את הקצב, ולכן צריך לזהות בעיות בשלב הפיתוח ולא בפרודקשן.

11. Skai: איש DevOps אחד וסוכן

Aviv Karbol, DevOps Architect ב-Skai, הציג את החלק היחיד בהרצאה שאינו מצגת מוצר. Skai — חברה בת 20 שנה בעולם ה-marketing וה-commerce media — בנתה פלטפורמת Omnichannel שדרכה עוברים למעלה מ-7 מיליארד דולר תקציב פרסום ומעל מיליון קמפיינים. המוצר החדש שהוצג, Studio, הוא "Agent Operation System עבור עולם המרקטינג": הלקוחות מייצרים סוכנים, כלים ו-data, בונים תהליכים חכמים ומשתפים אותם בארגון [30:53, 32:26].



המספרים שקדמו לשאלה שהוצגה לקהל — היקף הפיתוח של Studio בגרסה הראשונה.

מרעיון לגרסה ראשונה: 90 ימים, 6 ריפוזיטוריז, 15 אלף commits, מעל 4,000 pull requests, ו-64,042 הרצות CI (מהשקף; בדיבור נאמר "64 אלף workflows"). 100% מהקוד נכתב באמצעות AI — "עכשיו ה-AI אצלנו היא לא המלצה אלא הנחיה" [32:26]. במקביל נדרשה תשתית חדשה שלא הורצה קודם בפרודקשן — AgentCore ו-Temporal — ותהליכי דיפלוימנט שנבחרו במכוון לצאת מהתבניות המקובלות.

אז הגיעה השאלה לקהל: כמה אנשי DevOps הייתם צריכים כדי לתמוך בזה?

> **Aviv Karbol, [33:58] —** אז דיבופס אחד, אני, ודיבופס אייג'נט

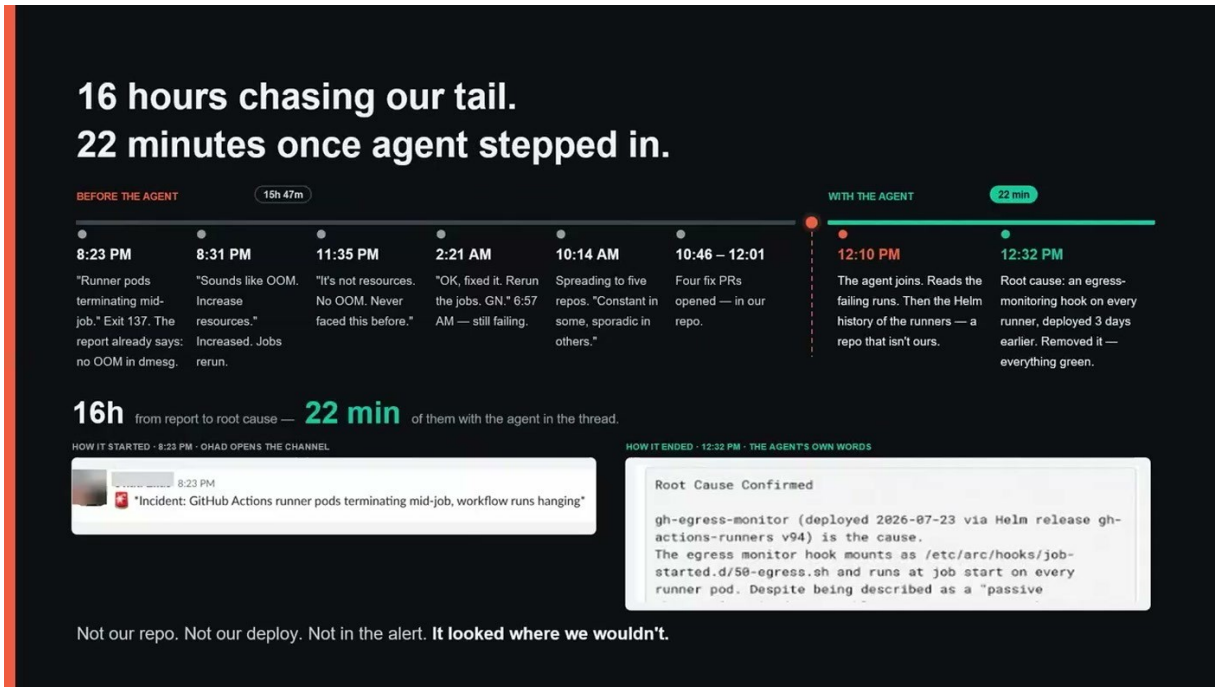
למה דווקא DevOps Agent

ההחלטה הוצגה כפרגמטית: "לא רצינו לקחת את זה ולייצר עוד SRE agent — כולם מייצרים SRE agent, וזה להתעסק ולבנות את הסוכן ולבנות את האינטגרציות. וחיפשנו פלטפורמה" [33:58]. DevOps Agent סיפק פלטפורמה מחוברת ל-Datadog ול-GitHub עם מנוע חקירה מוכן — "ומבחינתנו זו הייתה הבחירה הקלה".

הטיפים שניתנו מהשטח

- **IAM Role ממוקד.** לתת לסוכן את ה-scope הרלוונטי בלבד — הרסורסים ב-AWS שקשורים למוצר שלכם, לא יותר [33:58].
- **חקירות שרצות מאחורי הקלעים.** Datadog מטריג את הסוכן ברגע שיש alert, והסוכן מתחיל לעבוד לפני שמישהו פותח את הלפטופ [33:58].
- **Skills של PR Review.** ללמד את הסוכן על אילו practices להסתכל — האם מישהו נגע ב-Infrastructure as Code או באזורים שקשורים לתשתית [35:38].
- **בונים skill אחרי כל חקירה.** "החקירות הראשונות היו טובות, אבל הן לא היו 100%. ובכל פעם שהייתה חקירה נוספת, ראיתי איפה אנחנו צריכים להשלים את הסקיל הבא, כדי שהחקירה הבאה תתחיל מנקודה טובה יותר" [33:58].

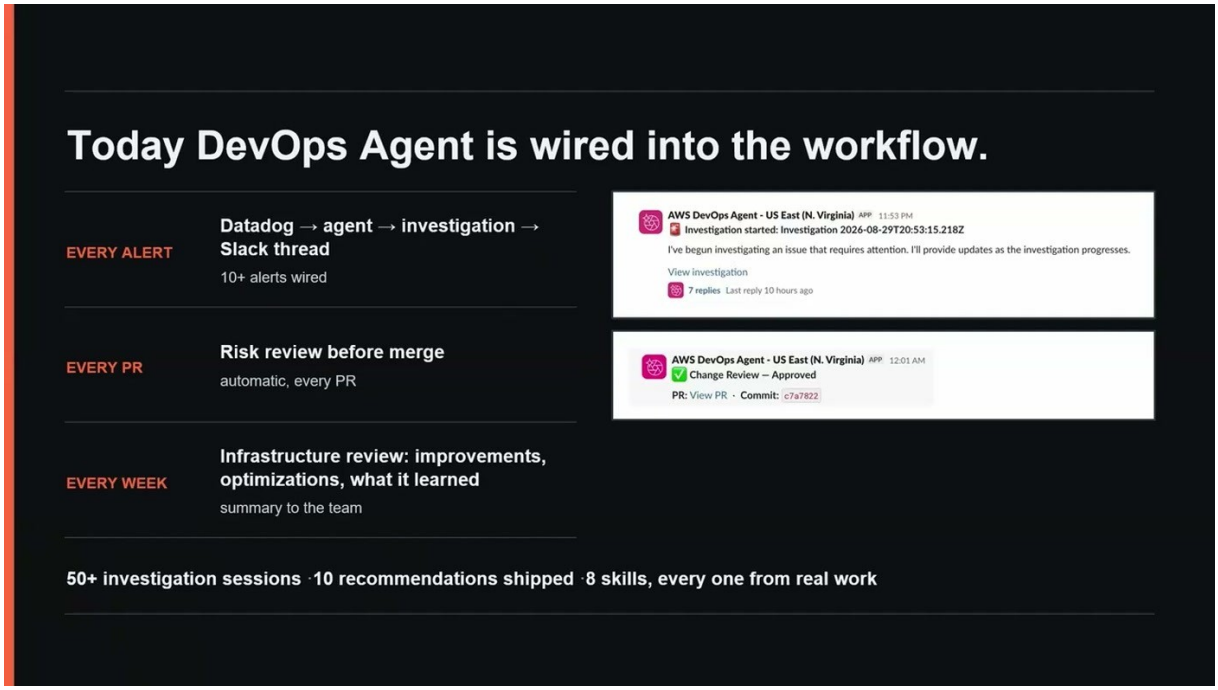
האירוע שסגר את הטיעון



ציר הזמן של האירוע האמיתי: שש נקודות עצירה של הצוות לאורך הלילה, מול שתי נקודות אחרי שהסוכן הצטרף.

"16 שעות שהצוות מחפשת את הבעיה" [35:38]. ההודעה נפתחה ב-8:23 בערב בערוץ incident; דרך ניחוש OOM, הגדלת משאבים, הרצה מחדש של jobs, והתפשטות לחמישה ריפוזיטוריז — עד ארבעה fix PRs שנפתחו. ב-12:10 הוכנס DevOps Agent לתמונה.

מה שקרה אז: הסוכן הכיר את הטופולוגיה, זיהה את הריפוזיטוריז הקשורים, וקישר את הבעיה ל-CI hook שהוכנס שלושה ימים קודם — ברפוזיטורי שאינו של הצוות. "וב-22 דקות הבעיה נפטרה" [35:38]. המשפט שסוגר את השקף: Not our repo. Not our deploy. Not in the alert. It looked where we wouldn't.



המצב הנוכחי ב-Skai: שלושה טריגרים קבועים — כל alert, כל PR, וסקירה שבועית — והמספרים שהצטברו.

כיום, לפי השקף והדברים: Datadog מייצר את האלרטים הנבחרים ומטריג ישירות את DevOps Agent, שפותח thread ב-Slack שבו רצה כל החקירה; כל PR עובר risk review אוטומטי לפני merge; ואחת לשבוע מגיעה סקירת

תשתית עם שיפורים ואופטימיזציות. המספרים על השקף: 50+ שני חקירה, 10 המלצות שיושמו, ו-8 — skills every one from real work"".

הצעד הבא שהוגדר: "אנחנו רוצים שהצוות יוכל להטריג אותו מה-Slack, אנחנו רוצים שהצוות יוכל לקרוא לו מ-GitHub... בעצם, אנחנו רוצים שה-DevOps יהיה ממש כמו קולגה, וזאת הסימביוזה שהיינו רוצים להגיע אליה" [37:11].

12. תמחור

מודל העלות: תשלום לפי זמן עבודה מצטבר, free trial, וקרדיטים חודשיים לפי תוכנית התמיכה.

- **Pay per use על זמן עבודה בפועל.** "אתם משלמים על כל שנייה שבה הסוכן מבצע חקירה, על כל שנייה שאתם בצ'ט מולו. כשהוא לא עובד, אתם לא משלמים עליו" [38:44]. לפי השקף, יכולות Release Management ב-preview זמינות ללא עלות נוספת.
- **Free trial של חודשיים.** בהרצאה נאמר שהוא כולל כ-55 שעות של תפעול הסוכן ללקוחות חדשים [38:44]; השקף מציין free trial hours ללא ציון המספר. מעבר למכסה — תמחור pay-as-you-go רגיל.
- **קרדיטים חודשיים ללקוחות תמיכה.** לפי השקף, הקרדיטים מתאפסים כל חודש (use it or lose it): Unified Operations — 100% מהוצאת ה-Support נטו של החודש הקודם; Enterprise Support — 75%; Business Support+ — 30%.

מה זה אומר בפועל משמעות מודל הקרדיטים: ארגון שכבר משלם על Enterprise Support או Unified Operations מקבל את רוב או כל השימוש בסוכן מכוסה מראש. כפי שנוסח בהרצאה — "אם אתם נמצאים כבר בתוכניות האלה, בוודאי ובוודאי אין תירוץ לא להשתמש ב-DevOps Agent" [38:44].

13. מה לוקחים מכאן

- **הערך נובע מהחיבורים, לא מהסוכן.** כל ההבדל בין חקירה טובה לחקירה חסרת ערך הוא במה שחיברתם: טלמטריה, ריפוזיטוריז, pipelines. בלי code repositories הסוכן לא היה מגיע לקומיט — לא בדמו של AWS ולא באירוע של Skai.
- **הטמעה היא תהליך מצטבר, לא הפעלה.** שתי העדויות מצביעות לאותו כיוון: החקירות הראשונות בינוניות, וכל חקירה מייצרת את ה-skill הבא. מי שמצפה לתוצאה מלאה בשבוע הראשון יתאכזב.
- **Scope ההרשאות הוא ההחלטה הראשונה.** ה-IAM Role שנותנים ל-Agent Space קובע גם מה הסוכן רואה וגם מה הוא לא. ההמלצה שניתנה מהשטח היא scope צר וממוקד למוצר, לא גישה רוחבית.
- **המיטיגציה עדיין בידי אדם.** בכל הדמואים הסוכן בנה תוכנית מפורטת, אבל הביצוע — rollback, re-run, פריסה מחדש — נעשה ידנית. זו החלטת עיצוב, וכדאי לשמור עליה בסביבות רגישות.
- **הכיוון הוא shift left.** Release Readiness Review, Release Testing ו-Headless mode כולם מזיזים את המוצר משכבת התפעול לשכבת הפיתוח — אל ה-IDE, ה-PR וה-pipeline. ארגון שמעריך את הכלי רק לפי MTTR מודד את החלק המצטמצם שלו.
- **נקודת כניסה זולה לבדיקה.** בין ה-free trial לקרדיטים של תוכניות התמיכה, עלות ה-POC נמוכה. הנחת המוצא הסבירה: לבחור שירות אחד עם טלמטריה טובה, לחבר אליו repo ו-pipeline, ולמדוד על פני חודש.

משאבים שהוצגו

בסוף ההרצאה הוצגו קישורים ל-user guide של DevOps Agent, לסדנה (workshop) שמאפשרת להתקין אותו ולהתנסות במגוון תרחישים, ולדמואים מוקלטים ב-YouTube. כתובת המוצר שהופיעה על השקף: aws.amazon.com/devops-agent [38:44, 40:15].

מילון מונחים

מונח	מה זה בהקשר של הסשן
Agent Space	מרחב העבודה של הסוכן. מגדירים בו גישה לחשבונות ענן דרך IAM Role, אינטגרציות, ואת ה-skills וה-memories שנצברו.
Application topology	גרף שהסוכן בונה ומתחזק אוטומטית — הישויות בסביבה וההקשרים ביניהן, כולל מיפוי של log groups, מטריקות ודיפלוימנטים לישויות.
Frontier Agent	המונח שבו AWS מתארת את הסוכן: פועל אוטונומית מקצה לקצה ולא רק מריץ פעולה בודדת לפי בקשה.
Instructions / AGENT.md	שכבת ידע שנטענת תמיד לכל אינטראקציה. מתאימה למדיניות ארגונית רוחבית.
Skills / SKILL.md	ידע פרוצדורלי שנטען לפי התאמה למשימה. Learned Skills נבנים אוטומטית; Custom Skills הם רנבוקים ו-SOP של הארגון.
Memories	עובדות שנשלפות לפי רלוונטיות. Managed — הסוכן מייצר לבד; User Authored — נאמרות לו בצ'אט.
RCA — Root Cause Analysis	שלב החקירה שמחזיר את מקור הבעיה בפועל, לצד תוכנית מיטיגציה.
MTTR	Mean Time to Resolution — הזמן הממוצע מזיהוי תקלה עד פתרונה.
Headless mode	הפעלת הסוכן ללא ממשק משלו, דרך MCP דו-כיווני ו-Agent-to-Agent Protocol, מתוך IDE או pipeline.
MCP — Model Context Protocol	פרוטוקול חשיפת כלים וקונטקסט לסוכנים. כאן משמש גם לחיבור מקורות טלמטריה וגם לגישה חיצונית לסוכן.
A2A — Agent-to-Agent Protocol	פרוטוקול לתקשורת בין סוכנים; מאפשר לסוכני קידוד לקרוא ל-DevOps Agent.
Generate agent spec	כפתור שמייצר פרומט מוכן מהמלצה, להעברה לכלי פיתוח (Kiro, Claude, Cursor) שיבצע את השינוי בקוד.
Release Readiness Review	בדיקה בזמן כתיבת הקוד של ההשפעה הצפויה על הפרודקשן כפי שהוא רץ בפועל — dependencies, מדיניות, access control.

מונח	מה זה בהקשר של הסשן
Triage Agent	תת-סוכן שמקשר תקלות מרובות לאירוע אחד ומסמן אותן כ-Linked, כדי למנוע חקירות כפולות.