Prodo-Typing: פרוטוטייפינג מהיר על AWS עם Builder Agents

AWS Summit Tel Aviv 2026 ,סיכום סשן — TLV-DEM309

Roy Osherove, Sr. 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-DEM309 מתוך AWS Summit Tel Aviv 2026, בהתבסס על ההקלטה המלאה שפורסמה באתר הסאמיט. הוא נבנה מתמלול אוטומטי של ההקלטה ומהסליידים שחולצו מהווידאו עצמו, כך שאפשר לחזור לתוכן בלי לצפות שוב בעשרים ושש הדקות. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** ארבע דקות קריאה. מה הוצג, ומה התביעה המרכזית של הסשן.
- **פרקים 2–3 — הבעיה והתבנית:** למה פרוטוטייפינג נתקע, ומהי תבנית Agent-Owned Account.
- **פרק 4 — שלושה דמואים:** שלוש אפליקציות שנבנו בשיטה הזאת, עם הארכיטקטורה וזמני הבנייה שנמסרו מהבמה.
- **פרקים 5–6 — הכלים:** LowKey ו-Kiro Crew, והדמו החי שרץ על הבמה.
- **פרק 7 — מה לוקחים מכאן:** מסקנות, מגבלות, ונקודות להמשך.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך שמות ומונחים לועזיים הופיעו בשיבוש פונטי ותוקנו ידנית מול הסליידים (למשל "אג'נט קור" ← AgentCore, "לוקי" ← LowKey, "קירו קרו" ← Kiro Crew). הציטוטים אומתו מול ההקלטה בחותמת הזמן המצוינת, והם מובאים כלשונם בתמלול — כולל שיבושי ASR. מספרים וזמני בנייה הם כפי שנמסרו מהבמה בעל פה או כפי שהופיעו על הסליידים; לא בוצעה כל אימות עצמאי שלהם.

1. תקציר מנהלים

שקופית הפתיחה: Full Stack Builder Agents & Agent-Owned Accounts — שתי התבניות שהסשן כולו נבנה סביבן.

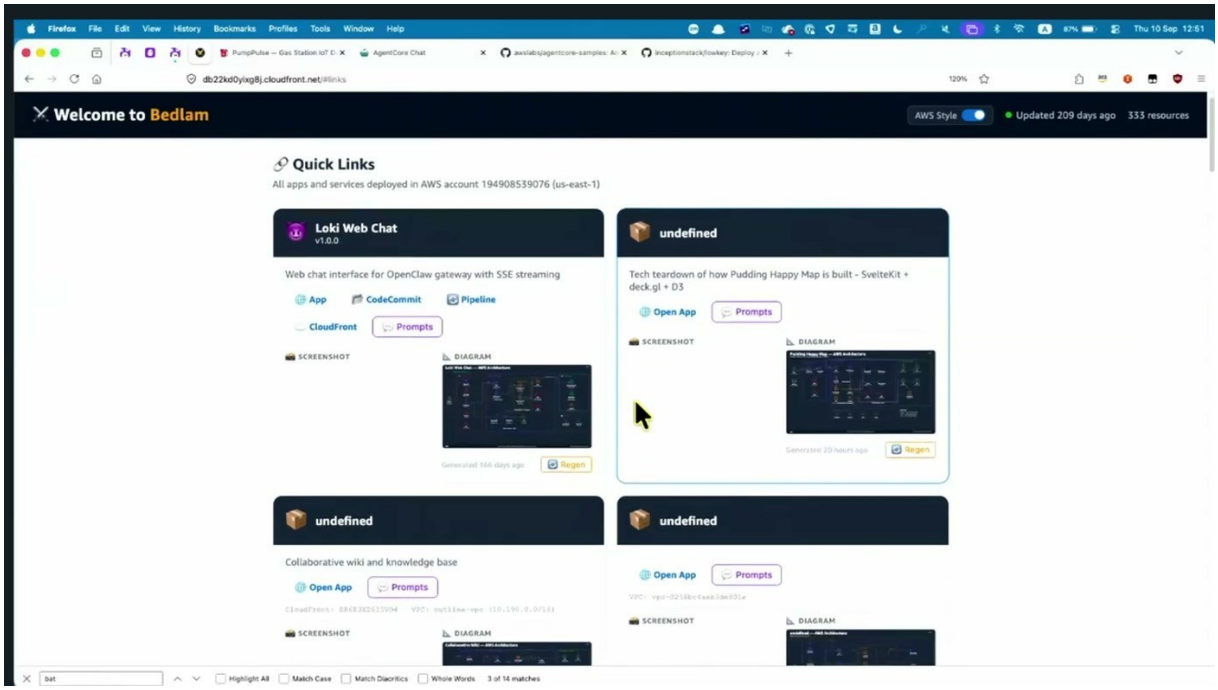
הסשן פותח בשלוש שאלות לקהל: מי השתמש בכלים כמו Lovable או Base44, מי משתמש בהם כחלק מהפרוטוטייפינג הארגוני, ומי מנסה לבנות פלטפורמה פנימית במקומם כדי לקבל יותר שליטה. השאלה השלישית היא הנושא: "זה יהיה הנושא שלנו היום איך אתם יכולים לבנות את ה-base 44 או את ה-louable שלכם ה-"AWS [0:00].

- **הטענה המרכזית: מחוללי אפליקציות מהירים אבל ללא שליטה על הארכיטקטורה.** המרצה מציג את Builder Agents כדרך לקבל את המהירות של Lovable ו-Base44 — "אמנם לא את ה-UI Design Experience אבל את המהירות אבל עם הכוח של [3:02] "AWS — כלומר פרוטוטייפ שמתפרס על כל קטלוג השירותים של AWS.
- **המנוף הוא לא המודל אלא ההרשאות.** הצעד שפותח את הכל, לפי המרצה, הוא מתן הרשאות דיפלוימנט לסוכן — אבל לא לסביבות קיימות: "נתנו לו AWS אקאונט חדש משלו" [6:06]. תבנית זו נקראת Agent-Owned Account.
- **גבול האבטחה עובר ברמת החשבון, לא ברמת המכונה.** "מבחינת איפה הסקיוריטי באונדרי שלנו הוא ברמת האקאונט ולא ברמת ה-[7:45] "VM. בחשבון הסנדבוקס אין דאטה של לקוחות ואין דאטה של פרודקשן, ולכן פריצה של הסוכן מגבולות ה-EC2 מסתכמת, לדבריו, בעלות תקציבית.
- **המדדים שנמסרו מהבמה: דקות ושעות, לא שבועות.** אפליקציית צ'אט מלאה מעל Bedrock AgentCore — "זמן היצירה של הדבר הזה מפרומט ועד שהדבר הזה רץ, 25 דקות" [10:50]. דשבורד IoT בזמן אמת עם Kinesis, Flink ו-TimescaleDB — "שלוש וחצי שעות" [12:21], כשחצי מהזמן המרצה היה ברכבת.
- **ה-deliverable הוא Infrastructure as Code, ולכן יש מסלול יציאה.** הסוכן כותב גם את התשתית כקוד (Terraform / CDK / CloudFormation), ולכן המעבר מחשבון הפרוטוטייפינג לסביבות אחרות — מה שהמרצה מכנה graduation — הוא לקיחת ה-IaC ודיפלוי במקום אחר [7:45].
- **החסם האמיתי הוא ארגוני, לא טכני.** "המון מה בוטלנק פה, הוא הפריקשן הפסיכולוגי ברמת חברה של לתת לאטנט את הפרמישן הנכונים כדי להאיץ תהליכים מסוימים" [15:23].

> **— [15:23]** אם יש דבר אחד שאני רוצה שתיקחו מהסשן הזה, זה אתם יכולים לקחת היום כל אייג'נט שיש לכם אם אתם תיתנו לו את הפרמישן הנכונים ואת הסביבה הנכונה אתם יכולים להרוד לעצמכם המון המון פריקשן.

2. הבעיה: הפער בין פרוטוטייפ לפרודקשן

המרצה מספר שנכנס ל-AWS שמונה חודשים לפני הסשן, ולפני כן היה פאונדר. כסולושן ארכיטקט הוא מקבל לפטופ, תקציב, טוקנים, ויכולת לפתוח חשבונות AWS "כפי יכולתך בשביל שאנחנו נוכל לעשות כל מיני פרוטוטייפים" [0:00]. מכאן נולד הניסוי: לקחת stateful agent, לתת לו חשבון AWS משלו, ללמד אותו לכתוב קוד ולתת לו סביבה שאליה הוא יכול לדפלט.



לוח ה-Quick Links שהסוכן עצמו ייצר: כל האפליקציות והשירותים שנפרסו בחשבון AWS אחד (194908539076, us-east-1), 333 משאבים. לפי המרצה, כל הפרויקטים ברשימה נוצרו בשבועיים הראשונים של העבודה בשיטה הזאת [1:31].

2.1 למה AWS מאט פרוטוטייפינג

הנקודה שהמרצה מדגיש היא שהאטיות איננה חוסר ידע: "אנשים שמבינים מה הם עושים ב-AWS, עדיין, לוקח הרבה זמן" [3:02]. הסיבה היא ריבוי הבחירות והחיכוך בהקמת שירותים חדשים.

> **— [3:02]** AWS זה כמו לגו, צריך לבחור המון דברים, יש המון פריקשן שמרימים סרוויסים חדשים וכולי, אז נושא הדיפלוימנט הוא פריקשן מאוד גדול

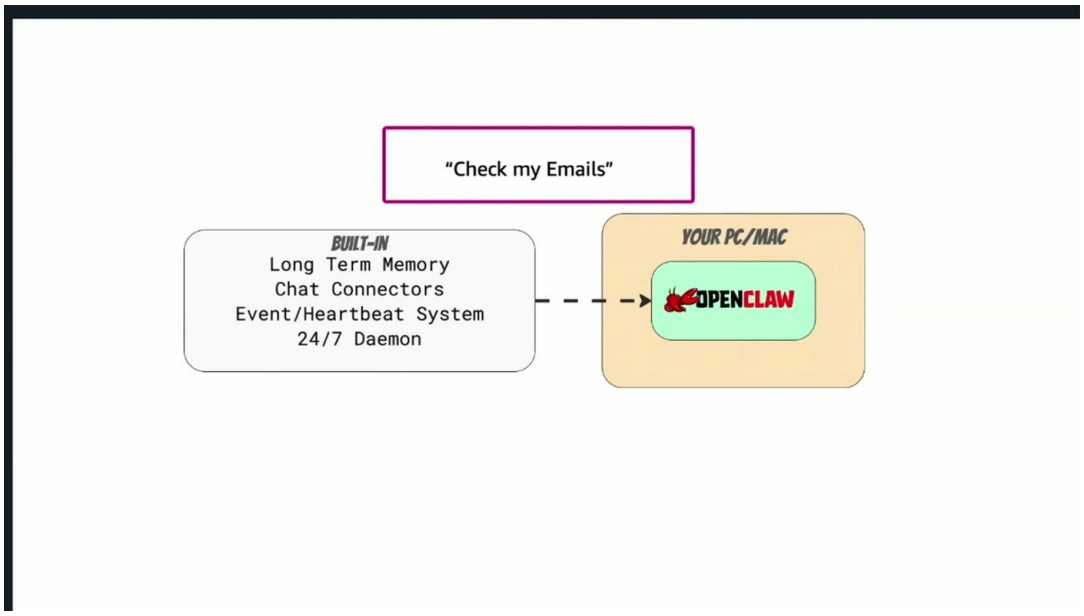
2.2 המעבר מ-inception ל-construction

המרצה מסייג במפורש את גבולות הדיון: הסשן אינו עוסק בפיתוח שוטף, טסטינג, דיפלוימנט ומוניטורינג בארגון, אלא בשלב הפרוטוטייפינג בלבד [3:02]. הכאב שהוא מתאר הוא רגע ה"פרודקטיזציה" — המילה שהוא מכנה "מאוד מאוד מאוד מעצבנת" — שבו מתברר שצריך לכתוב את הדבר מחדש: "הוא לא בסקייל, הוא לא בברנד, הוא לא בסקיוריטי, הוא לא כתוב כמו שצריך, אנחנו לא הונרים של הבקן, אין לנו שליטה על איך הדברים האלה עובדים, זה לא מתחבר ל-API עם הפנימיים שלנו, זה לא מתאים לרגולציה" [3:02].

ההצעה: להתחיל את הפרוטוטייפינג כבר על חשבונות ה-AWS של הארגון, עם חלק גדול מהדרישות האלה מובנות מראש, כך שהמעבר לשלב הבא יימדד בשעות או ימים "אבל בטח ובטח לא שבועות וחודשים" [4:33]. את השילוב הזה — פרוטוטייפ שנבנה מלכתחילה כמוצר — הוא מכנה בהמשך Prodo-Typing.

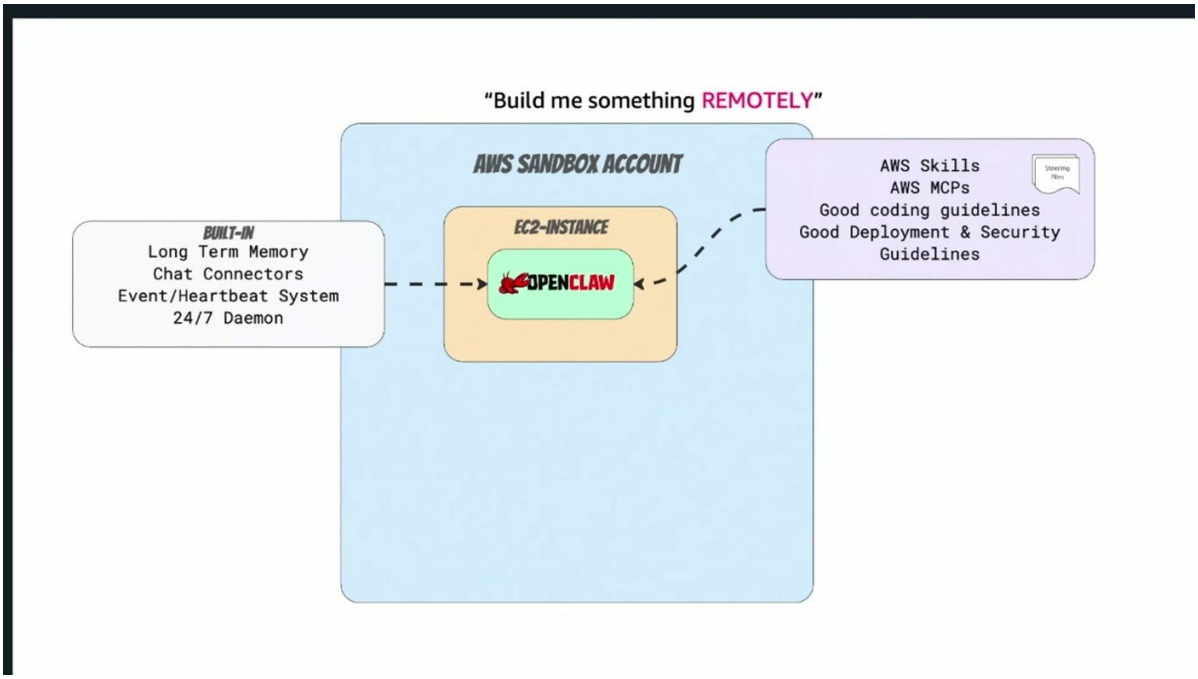
3. התבנית: מ-Local Agent ל-Agent-Owned Account

המרצה בונה את התבנית בשלושה צעדים, כשכל צעד מוצג כשקופית נפרדת. נקודת המוצא היא stateful agent מקומי — במקרה שלו OpenClaw, שאותו הוא מתאר כ"אחד הפרויקט Open Source הכי מצליחים שיש", ולדבריו כיום המוביל מבחינת stars ו-forks [4:33].



שלב א' — הסוכן רץ על המחשב האישי: זיכרון ארוך-טווח, מחברי צ'אט, מערכת אירועים/heartbeat ו-daemon שרץ 24/7. השימוש האופייני הוא עוזר אישי ("Check my Emails").

הצעד הראשון היה ללמד את אותו סוכן להיות Builder Agent: "נתנו לו AWS Skills, נתנו לו חיבור לכל AWS MCP [4:33]" Coding Guidelines, Deployment and Security Guidelines — כלומר החומרים שמכתיבים איכות קוד, נהלי דיפלוימנט ואבטחה, לצד היכולות המובנות של הסוכן.

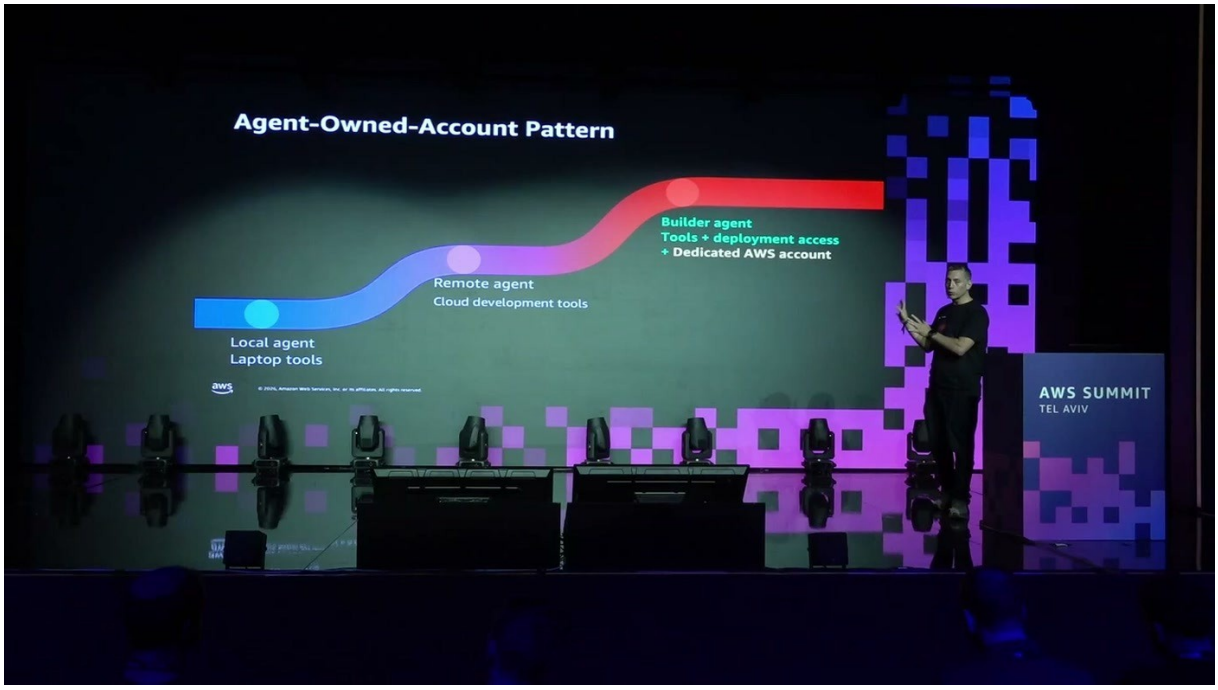


שלב ב' — אותו סוכן עובר ל-EC2 בתוך AWS Sandbox Account, בתוך VPC נפרד. השקופית מפרטת את מה שמוזרם אליו: AWS Skills, AWS MCPs, והנחיות קוד, דיפלוימנט ואבטחה.

לפי המרצה, ה-VPC הנפרד נועד "כדי לא לחשוף את הגייטווי שלו ולא לחשוף את ה-URL" [6:06]. אבל עד כאן, לדבריו, עדיין מדובר בסוכן מרוחק שיודע לכתוב קוד — משהו שרבים כבר מכירים מ-remote agents שמאזינים ל-PRs.

3.1 הצעד שפותח את הכל: הרשאות דיפלוימנט

כאן מגיע מה שהמרצה מכנה ה-unlock האמיתי. נקודת המפתח היא לא שהסוכן קיבל הרשאות, אלא היכן: "בטח שלא לסטייג'ינג בטח שלא ל-dev, בטח שלא ל-production. נתנו לו AWS אקאונט חדש משלו" [6:06]. הוא מזכיר שפתיחת חשבון AWS היא "קריאת API אחת" ולכן לא מכשול טכני.

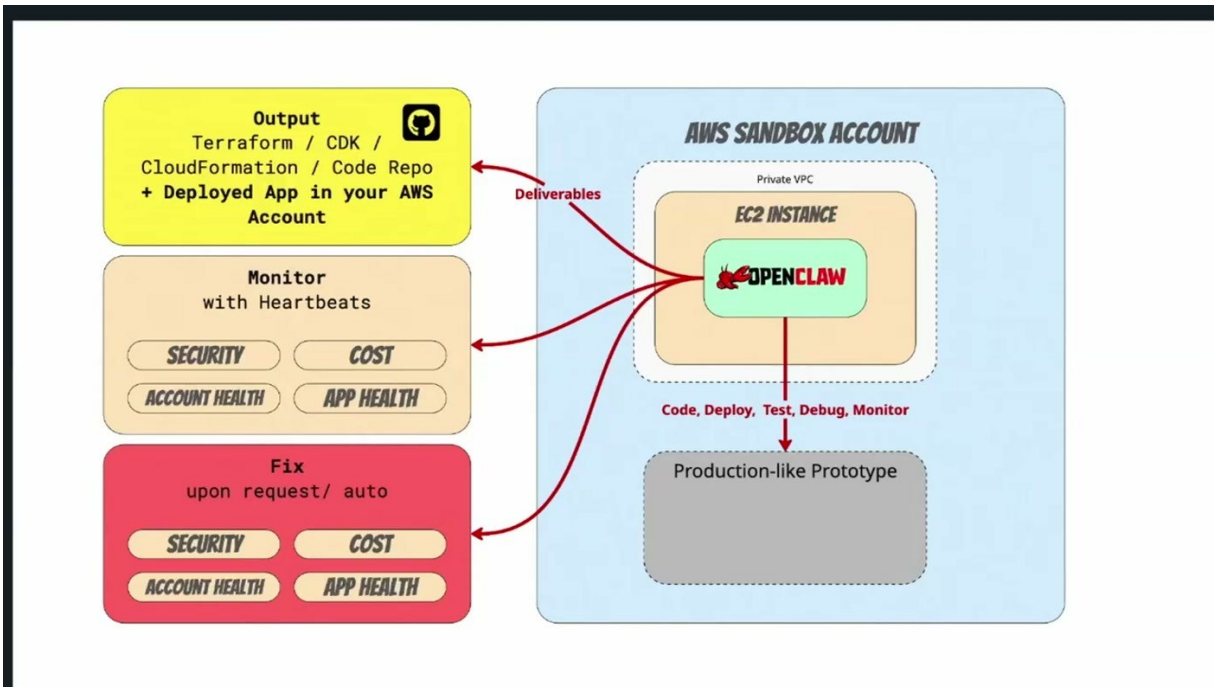


שלושת השלבים כעקומה: Local agent (כלי לפטופ) ← Remote agent (כלי פיתוח בענן) ← Builder agent (כלים + גישת דיפלוימנט + חשבון AWS ייעודי).

> **— [6:06]** אין שם קסטומר דאטה, אין שם פרודקשן דאטה, אין שם שום דבר אחר, מלבד הסרוויסים שהאג'נט שלי עושה להם דיפלוימן. למה זה חשוב? כי אין פה נושא של לפרוץ את הגבולות, אפילו אם האג'נט שלי פורץ את הגבולות של ה-EC2 instance, שלו המקרה הכי גרוע הוא בג'ט.

הצד השני של המשקולת, כדבריו, הוא תקציב: "רוצים לזוז מהר אנחנו נצטרך לטפל בנושא הבאדג'טים" [7:45]. הוא מקשר את הנקודה לסשן קודם באותו מסלול על ניהול תקציבי טוקנים וספנד, אך לא נכנס לפרטים מחוסר זמן.

3.2 מה הסוכן עושה בפועל, ומה יוצא מזה



התמונה המלאה של התבנית: הסוכן על EC2 בתוך חשבון הסנדבוקס עושה Code, Deploy, Test, Debug, Monitor; מולו — Output (Terraform / CDK / CloudFormation / Code Repo + האפליקציה הפרוסה), ניטור עם Heartbeats על ארבעה צירים (Security, Cost, Account Health, App Health), ותיקון לפי בקשה או אוטומטי על אותם ארבעה צירים.

הטיעון שמחזיק את הדיאגרמה הוא סימטריה של הרשאות: כל מה שהמפתח יכול לעשות בחשבון דרך CLI או API, הסוכן יכול לעשות מה-VM שלו. מכאן, לפי המרצה, אין צוואר בקבוק ארגוני — הסוכן מדפלט, מנטר, מדבג ובודק בעצמו [7:45].

> **— [7:45]** אם אני יכול לעשות את זה מהלפטופ שלי על האקאונט גם האג'נט יכול לעשות את זה מהVM שלו.

לולאת התיקון שהוא מתאר עוברת דרך כל הסטאק: "יש לי איזשהו בג באפליקציה הזאת, תבדוק מה קרה, האג'נט יכול להסתכל בלוגים, CloudWatch, CloudTrail, כל מה שצריך, להבין את סורס הבעיה, all the way down, בכל הסטק, ולשנות את הקוד, לדחוף את הקוד, לתקן אותו, לבדוק אותו עם Playwright. [9:16]" PlayWrite, כפי שיודגם בהמשך, הוא מנגנון האימות שהסוכן מפעיל על עצמו.

3.3 איכות התוצר היא פונקציה של מי שמנחה

זו הסתייגות שהמרצה מציב מיוזמתו, ושכדאי לקרוא אותה לפני שמאמצים את התבנית. הוא מנגיד שני משתמשים: פרודקט אונר לא-טכני שרוצה URL עובד לדמו יקבל תוצר שעובד — CloudFront, לוגין דרך Cognito — "אבל לא בכך הדבר הזה היא סקליבל וסקיור" [9:16]. לעומתו, ארכיטקטית שכותבת את הדרישות כארכיטקטית תקבל כמעט את אותה מהירות, בתוצר שהוא scalable ו-secure.

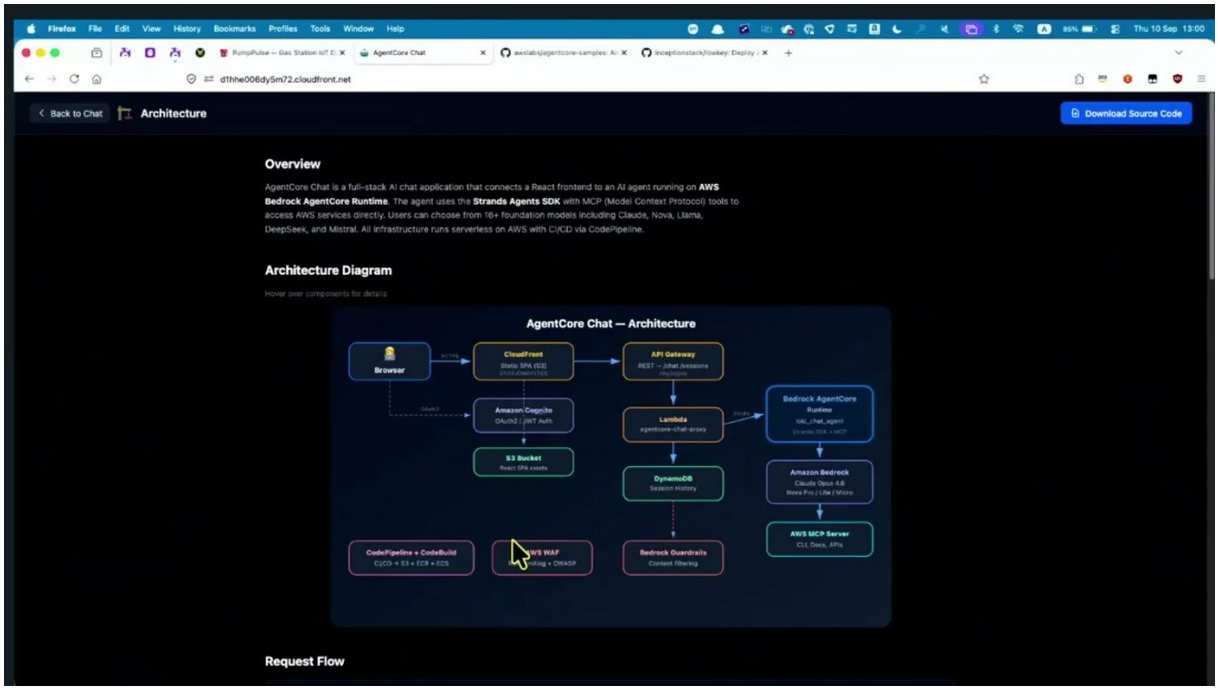
> **— [7:45]** אני גם אגיד שאיך שאני עובד עם האג'נט ואיך שאני בעצם נותן לו את ההוראות, זה מה שמחליט על האיכות של הקוד, אם זה well-architected, אם זה secure וכו'.

השליטה הזאת, לדבריו, נשארת בידי המנחה ברמת הפרומפט: "אני רוצה שיהיה לי לוד בלנסר פה ואני רוצה שיש לי פה לוגין... ופה שיהיה לי גרף דאטאבייס במקום whatever דיינמול... אבל אני יכול להחליט על הארכיטקטורה, יש לי שליטה מלאה וזה הכל ברמה של פרומפטים" [9:16]. זהו בדיוק הפער שהוא טוען שקיים מול Lovable ו-Base44.

4. שלושה דמואים שנבנו בשיטה

4.1 AgentCore Chat — 25 דקות

האפליקציה הראשונה שנבנתה כך נועדה, לדבריו, להוכיח איך Bedrock AgentCore עובד. הפרומפט כלל דרישות מוצר מפורשות: UI ובקאנד, שמירת היסטוריית שיחות ב-DynamoDB, בחירת סוכן, העלאת קבצים — והכל מאחורי לוגין [10:50].



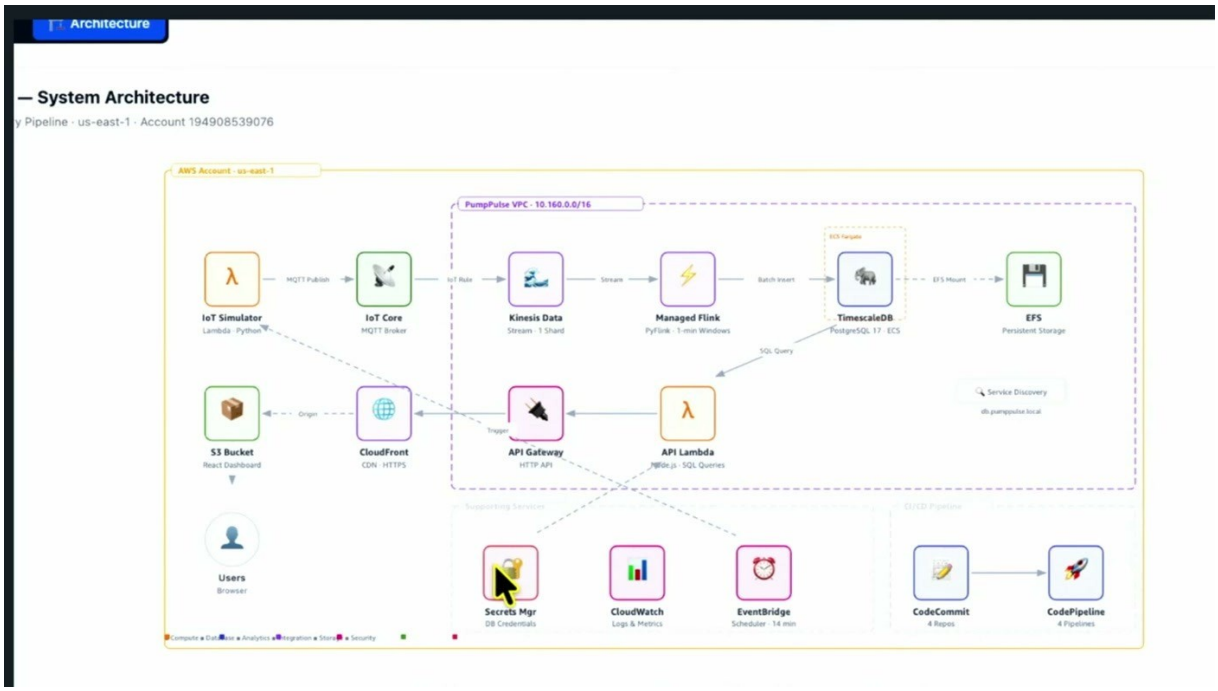
דף הארכיטקטורה שהאפליקציה מייצרת על עצמה: Browser ← CloudFront (S3 על SPA) ← Bedrock ← Lambda ← API Gateway, לצד Amazon Cognito ל-OAuth2/JWT, DynamoDB להיסטוריית משנים, AgentCore Runtime, AWS WAF, Bedrock Guardrails, שרת AWS MCP, ו-CodePipeline + CodeBuild ל-CI/CD.

לפי הטקסט בשקופית, הסוכן משתמש ב-Strands Agents SDK עם כלי MCP כדי לגשת לשירותי AWS ישירות, ומציע לבחירה 16+ מודלי יסוד. כל מה שנראה בדיאגרמה, מדגיש המרצה, נוצר על ידי אותו Builder Agent.

> **— [10:50]** זמן היצירה של הדבר הזה מפרומט ועד שהדבר הזה רץ, 25 דקות. 25 דקות. עוד לא דיברנו על האיכות של הקוד, אבל 25 דקות שהדבר הזה לייב.

4.2 PumpPulse — דשבורד IoT בזמן אמת, שלוש וחצי שעות

הדמו השני מורכב בהרבה: סימולציה של תחנות דלק ודשבורד שמציג בזמן אמת את מה שזורם במשאבות [12:21].



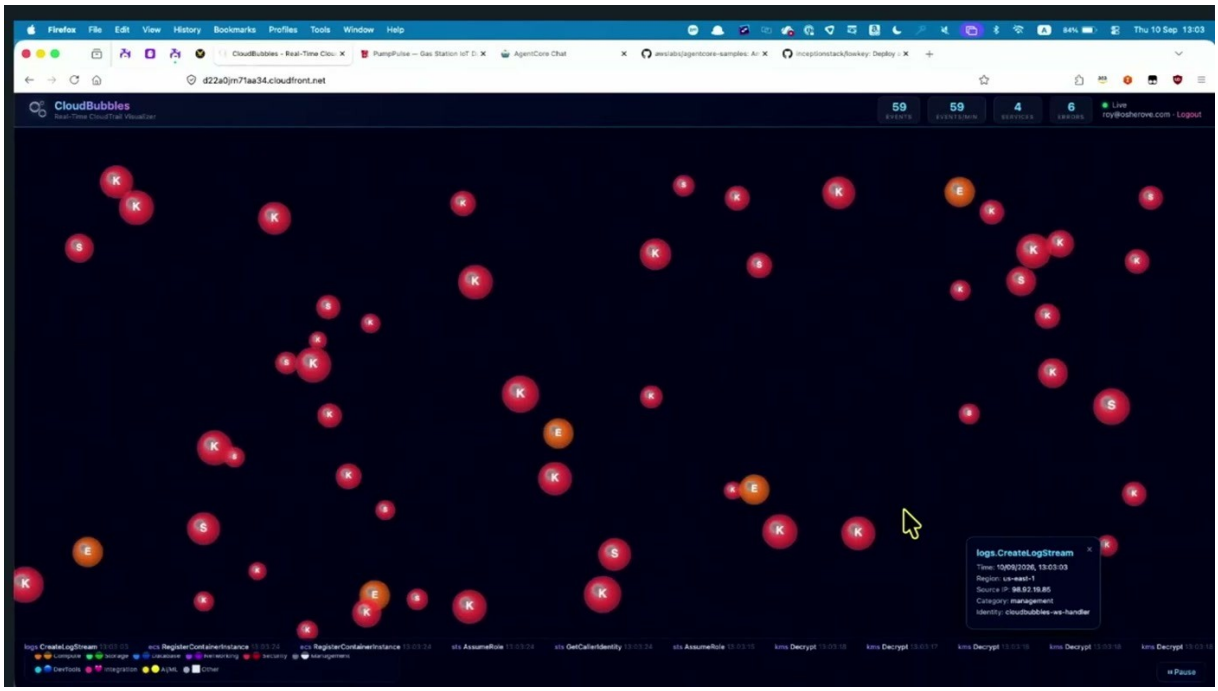
הצגרת המלאה: Lambda כסימולטור ← Managed Flink ← Kinesis Data Stream ← IoT Core (MQTT) ← IoT (חלונות דקה) ← TimescaleDB על ECS Fargate עם EFS; במקביל S3 ← CloudFront ← API Gateway ← API Lambda לדשבורד React, Secrets Manager לסודות ה-DB, ו-CodeCommit + CodePipeline כפייפליין פנימי.

הבחירה ב-CodeCommit ו-CodePipeline במקום GitHub הייתה מכוונת: "כי לא רציתי להסתמך על גיטאב, בכל זאת הדבר הזה נוצר בתוך AWS, ולא רציתי שיהיה לי דיפננצי חיצוני" [12:21]. זו בדיוק סוג ההחלטה הארכיטקטונית שהוא טוען שאי אפשר לקבל במחוללי אפליקציות חיצוניים.

— [12:21] זמן היצירה של האפליקציה הזאת, מפרומט ועד הדמו הזה שאתם רואים עכשיו, שלוש וחצי שעות. שלוש וחצי שעות, חצי מהזמן בכלל הייתי ברכבת.

4.3 CloudBubbles — ויזואליזציה של 25 דקות CloudTrail

הדמו השלישי נבנה, לדבריו, כמבחן גבול: "בואו נבנה משהו שבחיים לא הייתי יודע איך לבנות". התוצאה היא ויזואליזציה בסגנון Bubble Bobble שבה כל בועה היא אירוע CloudTrail בחשבון, בזמן אמת [13:52].



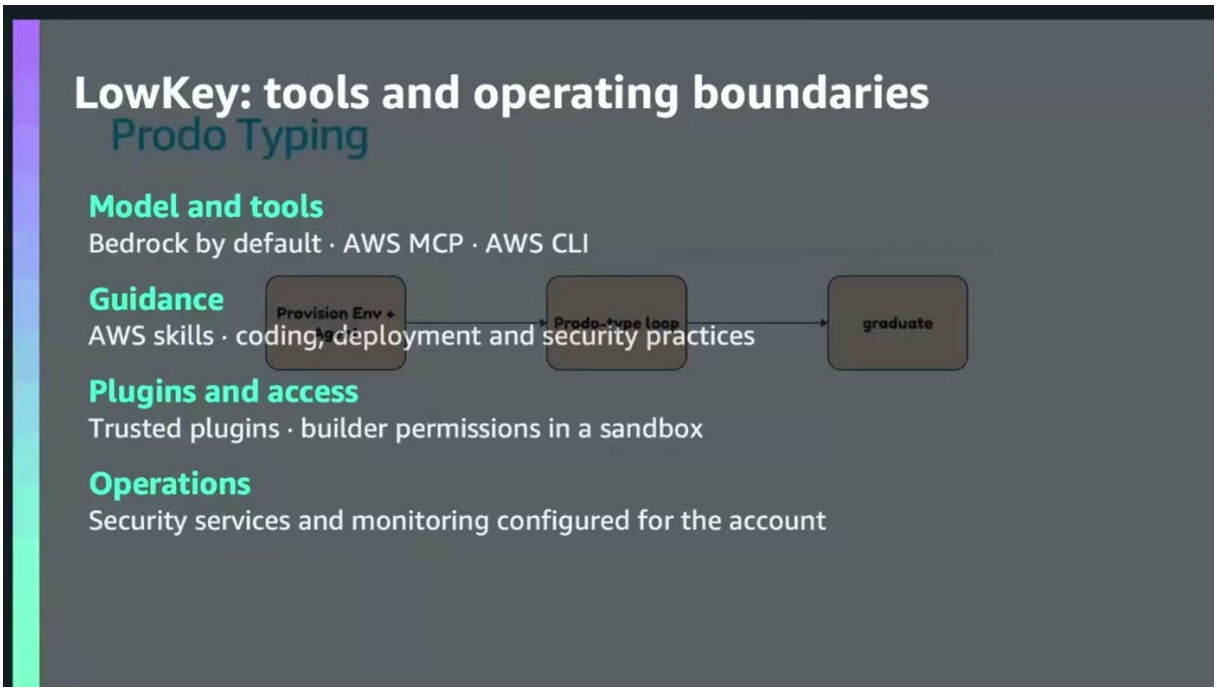
CloudBubbles — Real-Time CloudTrail Visualizer: 59 אירועים, 59 אירועים לדקה, 4 שירותים, 6 שגיאות. לחיצה על בועה חושפת את פרטי האירוע (logs.CreateLogStream, אזור, כתובת מקור, קטגוריה, זהות), ולמטה רץ פיד האירועים עם קידוד צבע לפי קטגוריית שירות.

> **[13:52] —** בחיים לא הייתי יודע לבנות כזה UI, ו בחיים לא הייתי יכול לבנות כזה פיצ'ר מתוך דברים כמו סרט פרטי לעבור בייס 44. זמן היצירה של הדבר הזה 25 דקות מהרעיון רעיון היה ב-2 בלילה דרך הטלפון דרך טלגרם במקרה הזה.

הפרט על הטלפון והטלגרם אינו צבעוני בלבד: הוא ההוכחה לכך שהסוכן הוא שירות מתמשך שרץ בחשבון ולא סשן על הלפטופ. זו התכונה שהופכת אותו, בלשון השקופיות, ל-"Stateful, Persistent Agent 24/7".

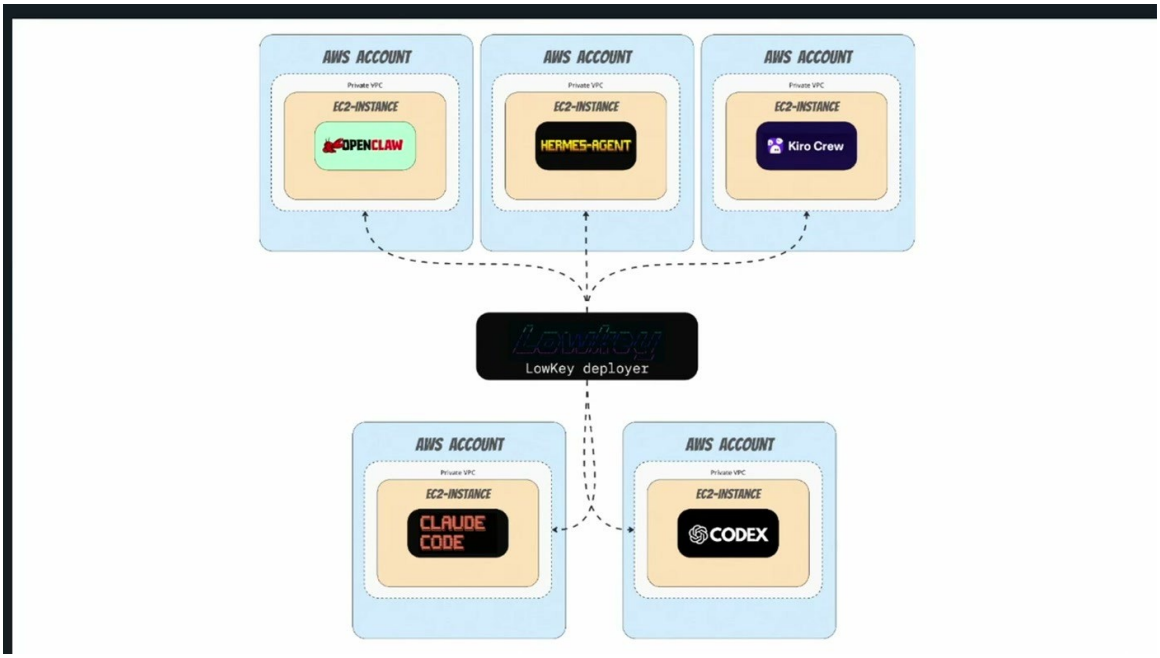
5. Prodo-Typing ו-LowKey

אחרי הדמואים המרצה נותן שם לתבנית. Prodo-Typing, בהגדרתו, הוא "production scale פרוטוטייפ" — היכולת לייצר פרוטוטייפ שהוא fully deployed, feature rich, scalable ו-secure, וקרוב לפרודקשן הרבה יותר מכל דבר קיים [15:23].



גבולות הפעולה של LowKey כפי שהוגדרו על השקופית: מודל וכלים (Bedrock כברירת מחדל, AWS MCP, AWS CLI), הנחיה (AWS skills) ופרקטיקות קוד/דיפלוימנט/אבטחה), פלאגינים וגישה (פלאגינים מהימנים, הרשאות builder בסנדבוקס), ותפעול (שירותי אבטחה וניטור מוגדרים לחשבון). למטה — לולאת העבודה: graduate ← Prodo-type loop ← Provision Env + Agent.

LowKey הוא כלי open source שהצוות פרסם. תפקידו צר ומוגדר: לפרוס stateful agent לתוך חשבון AWS — לכיוון Bedrock או EC2 — ולקנפג אותו מראש עם ההרשאות, הסקילס והפלאגינים הנכונים, כך שהמשתמש לא צריך להתמודד עם כאב הפריסה [15:23, 23:05].



LowKey deployer כשכבת פריסה אחת מול סוכנים שונים: OpenClaw, Hermes-Agent, Kiro Crew, Claude Code ו-Codex — כל אחד על EC2 בתוך VPC פרטי, בחשבון AWS נפרד משלו.

— [16:55] אז יש לכם סוג של Builder Agent in a Box לצורך העניין. זה כמו השלב הזה הראשון בחללית שהבוסטר רוקט שלוקח אותה לחלל, אבל הוא לא מגיע לחלל. זה רק הדיפלוימנט.

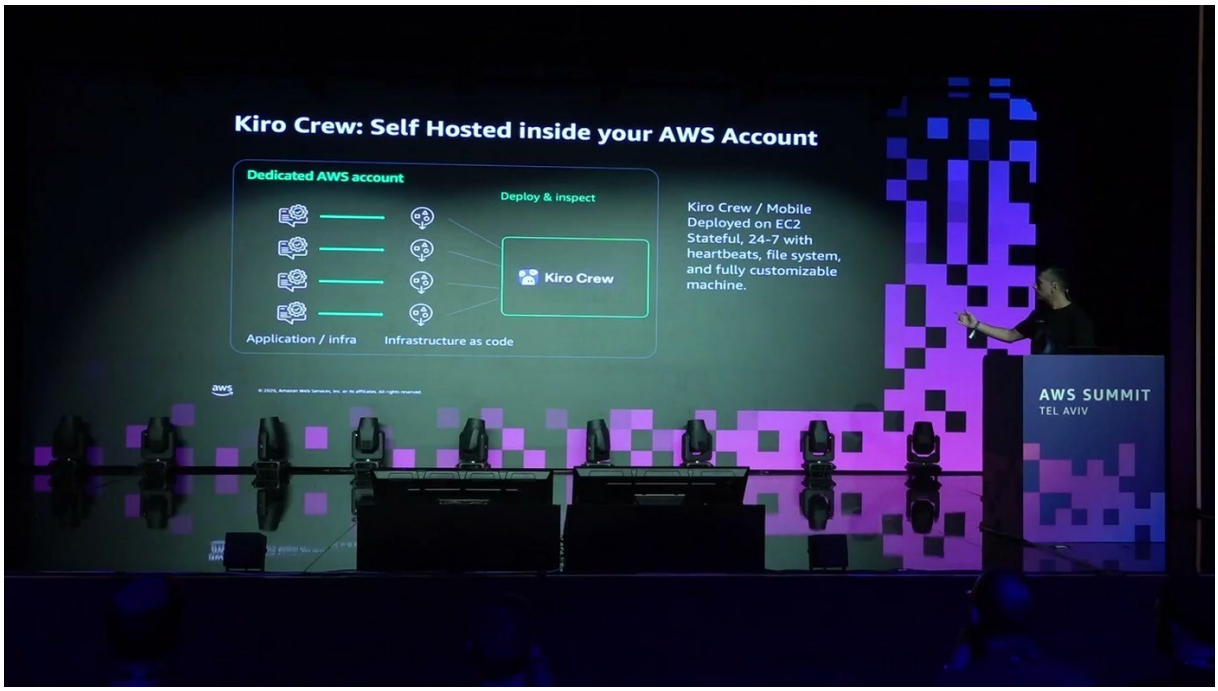
מודל העבודה שהוא מדמיין הוא זוגי: פרודקט אונר וארכיטקט/ית שעובדים יחד על אותו חשבון פרוטוטייפינג, "אחד אחרי על הוויז'ן, השני אחרי על הטקניקל, אבל ביחד הם יכולים לרוץ מאוד מאוד מהר ברמה של ימים ושעות" [16:55]. את יוזקייסים הוא מונה מחקר פנימי, אפליקציות, ו-DevOps/תשתית.

6. Kiro Crew — הגרסה המאושרת ארגונית

החלק האחרון של הסשן מוקדש ל-Kiro Crew, שהמרצה מציג כתשובה לארגונים שלא ירצו להריץ סוכנים כמו OpenClaw. הוא מבחין תחילה: Kiro הוא שירות נפרד שאפשר לצרוך בסבסקריפשן ואפילו בלי חשבון AWS, בעוד Kiro Crew הוא מוצר open source לחלוטין [18:30].

מיצוב Kiro Crew בקטגוריית ה-24/7 Stateful, Persistent Agents, לצד OpenClaw, Hermes, NanoClaw, NemoClaw ו-Grok Bot.

ארכיטקטונית, Kiro Crew הוא gateway שיושב מעל Kiro CLI ומדבר איתו ב-ACP. מתחתיו נדרש Kiro CLI מותקן ומופעל על מכונה; מעליו מגיעות היכולות: חיבור למובייל ולערוצים (Slack, Telegram, Discord), אורקסטרציה, sub-agents, ניהול זיכרון, knowledge graph ו-heartbeats [18:30].



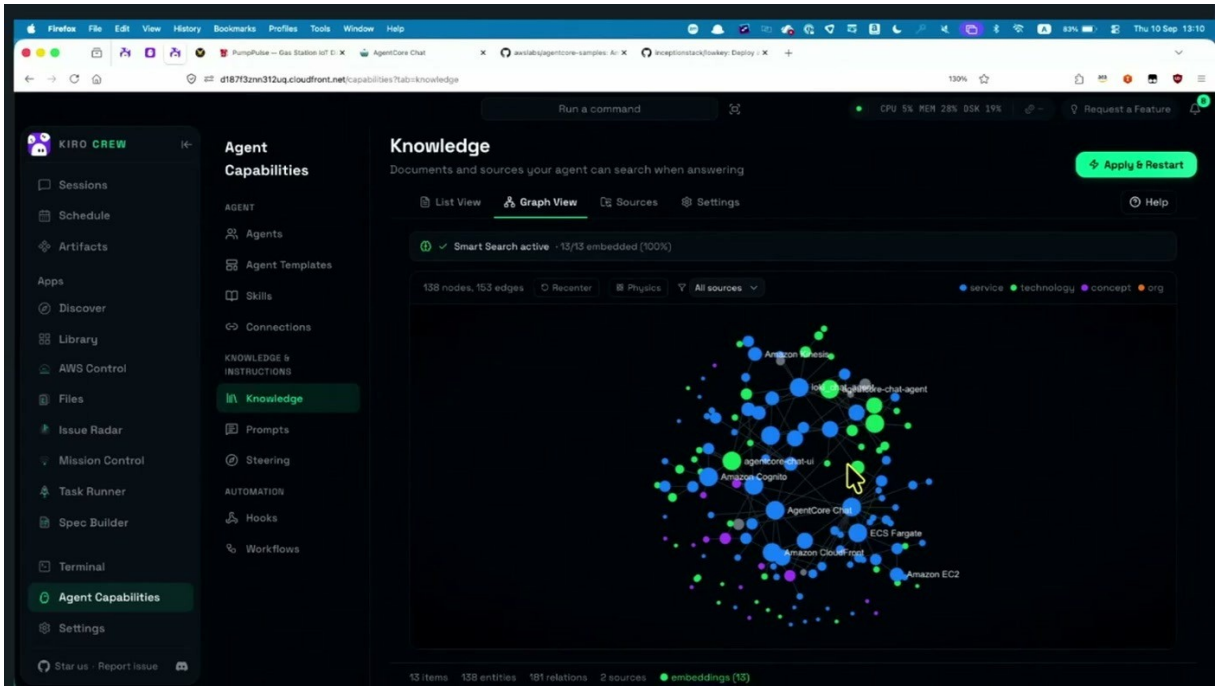
Kiro Crew self-hosted בחשבון AWS ייעודי: שרשראות של Application/infra ו-Infrastructure as code שנפרסות ונבדקות מול המכונה — Deployed on EC2, stateful, 24-7, עם heartbeats, מערכת קבצים ומכונה שניתנת להתאמה מלאה.

[18:30] — אבל אם היום הארגון שלכם פוחד מדברים כמו open cloud וכולי, קירו קרו הוא הרבה יותר secure by default, הוא עבר את ה-up-sec approval של AWS

המרצה מוסיף מגבלה אחת במפורש: Kiro Crew אינו תומך כיום ב-third-party providers, אך מאחר שהוא open source "אתם יכולים טכנית להוסיף את זה, שהוא דבר לא מונע" [20:01]. מבחינת פריסה — אותה תבנית: EC2 בתוך חשבון AWS, עם ההרשאות הנכונות, ומשם זהו Builder Agent לכל דבר.

6.1 סיור בממשק, והדמו החי

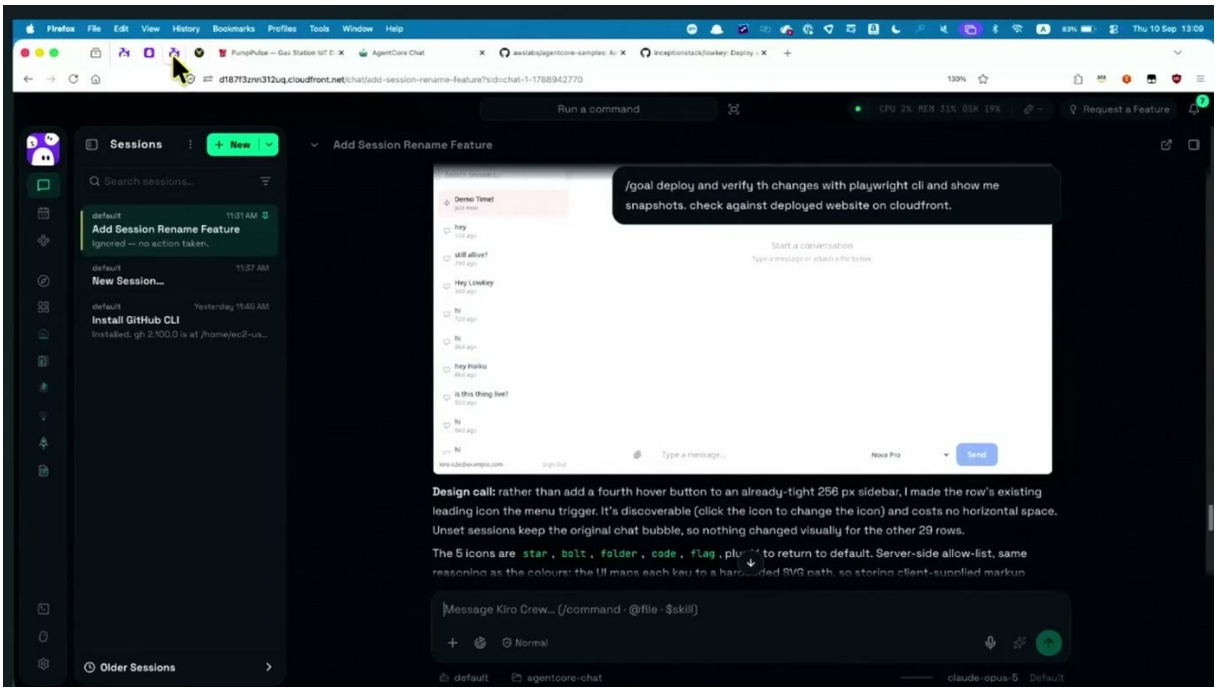
הסיור בממשק מכסה סשנים, Scheduled Actions (ה-heartbeats), Artifacts, טרמינל, וקסטומיזציה של יכולות הסוכן. אחד הפריטים שהמרצה מדגיש הוא ניהול הידע: הסוכן מחזיק knowledge graph, ואפשר להגדיר שמכל סשן הוא יוסיף אוטומטית את המסמכים והארטיפקטים לזיכרון [20:01, 21:33].



מסך Knowledge של Kiro Crew ב-Graph View: 138 nodes, 153 edges, 13 מתוך 13 מסמכים מוטמעים (100%), עם צמתים כמו Amazon Kinesis, AgentCore Chat, Amazon Cognito, ECS Fargate ו-Amazon EC2 — כלומר הידע שנצבר מהפרויקטים שנבנו בפועל.

לצד הזיכרון, הוא מראה נקודות שליטה ארגוניות: ייצוא וגיבוי של הזיכרון והעברתו בין סוכנים, קביעת steering ו-prompts, ו-custom hooks שמופעלים על כל tool use "אולי להפעיל איזשהו משהו של החברה שלי, כדי לוודא שזה עובר תחת איזשהו דברים מסוימים" [21:33]. הוא מציין גם חיבור ל-MCP Gateway כדי לזהות כלים שכבר רשומים ב-AgentCore Gateway.

הדמו החי שרץ על הבמה היה בקשה קטנה לשנות צבעי אייקונים בסרגל הצד של אפליקציית AgentCore Chat, במודל Claude Opus 5. שני מנגנוני ההרצה שהוא מדגים סביבה ראויים לתשומת לב ארגונית: Trust All Tools כדי שהסוכן לא יבקש אישור על כל פעולה, ו-YOLO mode שמחיל זאת על כל הסשנים.



הסשן החי ב-Kiro Crew: פקודת ה-goal שהוקלדה מהבמה — deploy ואימות השינויים מול האתר הפרוס ב-CloudFront באמצעות Playwright CLI — ולצדה הנמקת העיצוב שהסוכן החזיר בעצמו.

[23:05] אני יכול לכתוב פה גם slash goal, כמו make sure it works using playwright CLI after you are done coding it. ובעצם מה שיקרה עכשיו זה שאחרי שאני נותן goal כזה, אז יש פה איזשהו nudging cycle שמאפשר לאג'נט לתשל את עצמו ולהמשיך את הפיצ'ר הזה

הדמו לא הספיק להסתיים בזמן הסשן. בבדיקה האחרונה מהבמה הסוכן עדיין היה באמצע: "new icon color classes survive, tailwinds purge, deploying" [24:39]. המרצה סגר בהערה עצמית — "יכול להיות שהייתי צריך להחיל את הדמו קצת יותר מוקדם" — וזו נקודה כנה ששווה לזכור: לולאת ה-code-deploy-verify אורכת דקות, לא שניות.

7. מה לוקחים מכאן

- **התבנית עובדת עם כל סוכן שיש לכם כבר היום.** המרצה מדגיש שזו אינה דרישה למוצר חדש: כל agent שתתנו לו את ההרשאות הנכונות ואת הסביבה הנכונה הופך ל-LowKey. [15:23] Builder Agent ו-Kiro Crew הם קיצור דרך, לא תנאי.
- **הגבול הנכון הוא חשבון סנדבוקס ייעודי, לא VM.** לא dev, לא staging, לא production — חשבון חדש נטול דאטה של לקוחות. זהו הרכיב היחיד בסשן שעליו תלויה כל טענת האבטחה.
- **מי שמנחה קובע את התוצר.** פרודקט אונר יקבל דמו עובד; ארכיטקט/ית יקבלו תוצר scalable ו-secure באותה מהירות. אם בארגון רוצים לאמץ את זה, ההשקעה היא בסטנדרטים ובפרומפטים — לא בכלי.
- **ה-IaC הוא נכס היציאה.** כל עוד התוצר כולל Terraform או CloudFormation, ה-graduation לסביבה אמיתית הוא דיפלוי במקום אחר ולא כתיבה מחדש. בלי זה חוזרים בדיוק לכאב ה"פרודקטיזציה" שהסשן בא לפתור.
- **התקציב הוא הצד השני של המשקולת.** המרצה מודה שהוא לא הספיק להיכנס לנושא; ארגון שמאמץ את התבנית חייב לפתור ניהול תקציב וטוקנים לפני, לא אחרי.
- **הכשלים בסשן הם החסם הארגוני והדמו שלא סגר.** "הפריקשן הפסיכולוגי ברמת חברה" [15:23] הוא לדבריו הבוטלנק האמיתי; ולולאת האימות העצמי של הסוכן לא השלימה את עצמה בזמן הסשן.

לבדיקה לפני אימוץ המספרים בסשן (25 דקות, שלוש וחצי שעות) הם דיווח בעל פה של המרצה על פרויקטים שלו, לא מדידה מבוקרת, ואיכות הקוד לא נבחנה מעל הבמה — המרצה עצמו מציין "עוד לא דיברנו על האיכות של הקוד" [10:50]. Trust All Tools ומצב YOLO מסירים את אישור המשתמש לכל פעולת כלי; בסביבה מפוקחת זה שינוי מדיניות, גם כשהוא מתרחש בחשבון סנדבוקס.

8. מילון מונחים

מונח	מה זה בהקשר של הסשן
Prodo-Typing	המונח שטבע המרצה: פרוטוטייפ בקנה מידה של פרודקשן — fully deployed, feature rich, scalable ו-secure מהיום הראשון [15:23].
Builder Agent	סוכן שמקבל לא רק כלי קוד אלא גם גישת דיפלוימנט וחשבון AWS ייעודי, ולכן מסוגל לקודד, לפרוס, לבדוק, לדבג ולנטר [7:45].
Agent-Owned Account	התבנית המרכזית: חשבון AWS ייעודי בבעלות הסוכן, שבו הוא מריץ את כל הפעולות. גבול האבטחה עובר ברמת החשבון ולא ברמת המכונה [7:45].
Stateful Agent	סוכן מתמשך שרץ 24/7 עם זיכרון ארוך-טווח, heartbeats ומחברי צ'אט — להבדיל מסשן CLI חד-פעמי. דוגמאות מהשקופיות: OpenClaw, Hermes, NanoClaw, NemoClaw, Grok Bot, Kiro Crew.
LowKey	כלי open source שפורס stateful agent לתוך Bedrock או EC2 בחשבון שלכם, מקונפג מראש עם הרשאות, סקילס ופלאגינים מהימנים [15:23].
Kiro Crew	gateway open source מעל Kiro CLI (דרך ACP), שמוסיף מובייל/ערוצים, אורקסטרציה, sub-agents, זיכרון, knowledge graph ו-heartbeats. לפי המרצה עבר AppSec approval פנימי [18:30].
Heartbeats / Scheduled Actions	הרצות יזומות של הסוכן על לוח זמנים — הבסיס לניטור המתמשך של Security, Cost, Account Health ו-App Health שבשקופית התבנית.
Graduation	המעבר מחשבון הפרוטוטייפינג לסביבות אחרות, על בסיס ה-Infrastructure as Code שהסוכן ייצר [7:45].
YOLO / Trust All Tools	הגדרות ב-Kiro Crew שמבטלות את בקשת האישור לכל שימוש בכלי — ברמת הסשן או ברמת כל הסשנים [23:05].

9. מקורות והמשך

שקופית הסיום עם הקישורים שהוצגו מהבמה: Kiro בכתובת kiro.dev, LowKey ב-github.com/Inceptionstack/lowkey, ו-Prodo-Typing ב-robotpaper.ai/prodo-typing, לצד הזמנה לוורקשופ בן שלוש שעות.

בסיום הוזכר וורקשופ בן שלוש שעות ללקוחות, שבו מוקצית סביבה ומוקם Kiro, והמשתתפים בונים פרוטוטייפים משלהם [24:39]. המרצה הפנה לפנייה אליו במייל להצטרפות.