ECS בלי לנהל שרתים: Managed Instances ו-Express Mode

AWS Summit Tel Aviv 2026 ,סיכום סשן — TLV-SVS304

Evgeny Grinfeld, Senior Solutions Architect, AWS · Yaki Ratz, Senior Technical Account Manager, AWS · Hod Raz, Software Team Leader, Earnix

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

סיכום מאת קובי אלמוג

על המסמך הזה

המסמך מסכם את הסשן TLV-SVS304 מתוך AWS Summit Tel Aviv 2026, במסלול AWS for Containers and Platform Engineering. הוא נבנה מתמלול מלא של ההקלטה ומהסליידים שחולצו מהווידאו, כך שאפשר לחזור לתוכן בלי לצפות שוב בארבעים ואחת הדקות. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

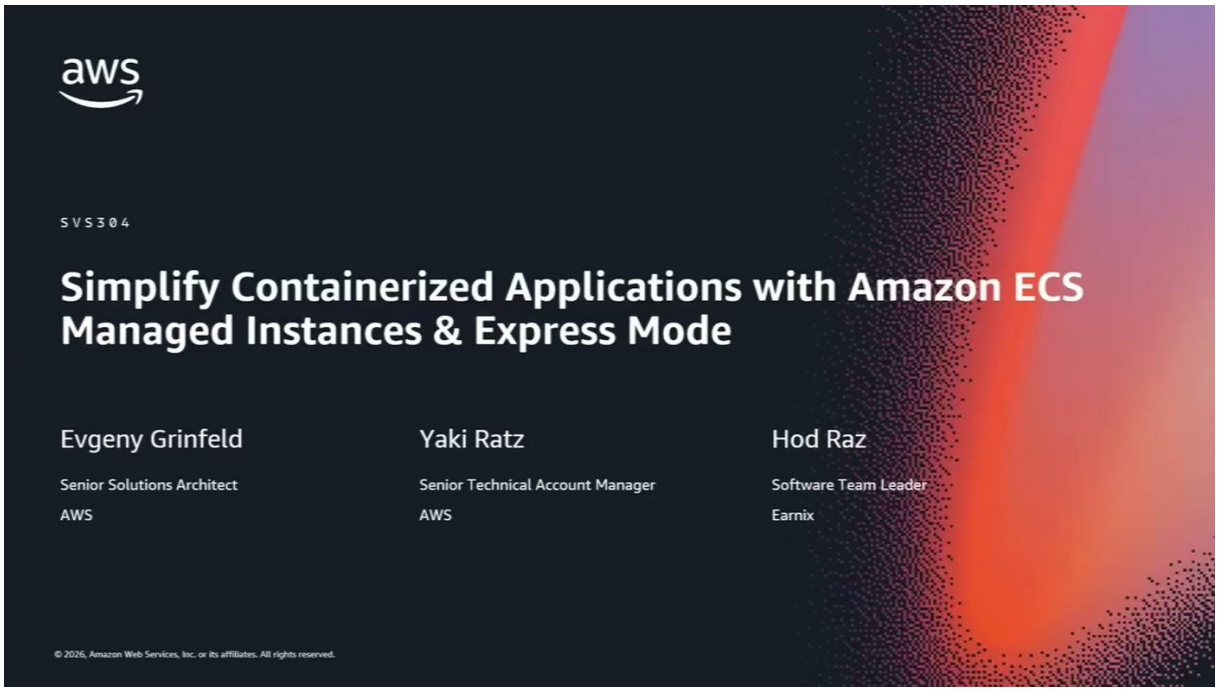
הסשן מחולק לשלושה חלקים, בהתאם לשלושת הדוברים: סקירת ECS ושני החידושים (Express Mode ו-Managed Instances), צלילה לאופטימיזציית עלויות ולמנגנונים שמאחורי Managed Instances, וסיפור לקוח מחברת Earnix.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה הוצג ומה רלוונטי להחלטה על data plane.
- **פרקים 2–4 — הרקע והחידוש:** מה זה ECS, מה Express Mode משנה בחוויית המפתח, ואילו אפשרויות Capacity קיימות.
- **פרקים 5–7 — Managed Instances לעומק:** היכולות, מודל האחריות והבידוד, ואופטימיזציית עלויות.
- **פרק 8 — מאחורי הקלעים:** קונפיגורציה, בחירת אינסטנס, החלפת שרתים ופאטצ'ינג.
- **פרק 9 — מקרה מהשטח:** איך Earnix מפיצה דאטה ב-BitTorrent על גבי ECS.
- **פרק 10 — מה לוקחים מכאן:** המלצות הדוברים ונקודות להמשך.

> **הערת מקור** המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך שמות ומונחים לועזיים מופיעים בו בשיבוש פונטי, והם תוקנו מול הסליידים. הציטוטים אומתו מול התמלול בחותמת הזמן המצוינת ומובאים כלשונם, כולל שיבושי ASR שנותרו. מספרים שמופיעים על הסליידים צוינו ככאלה.

1. תקציר מנהלים

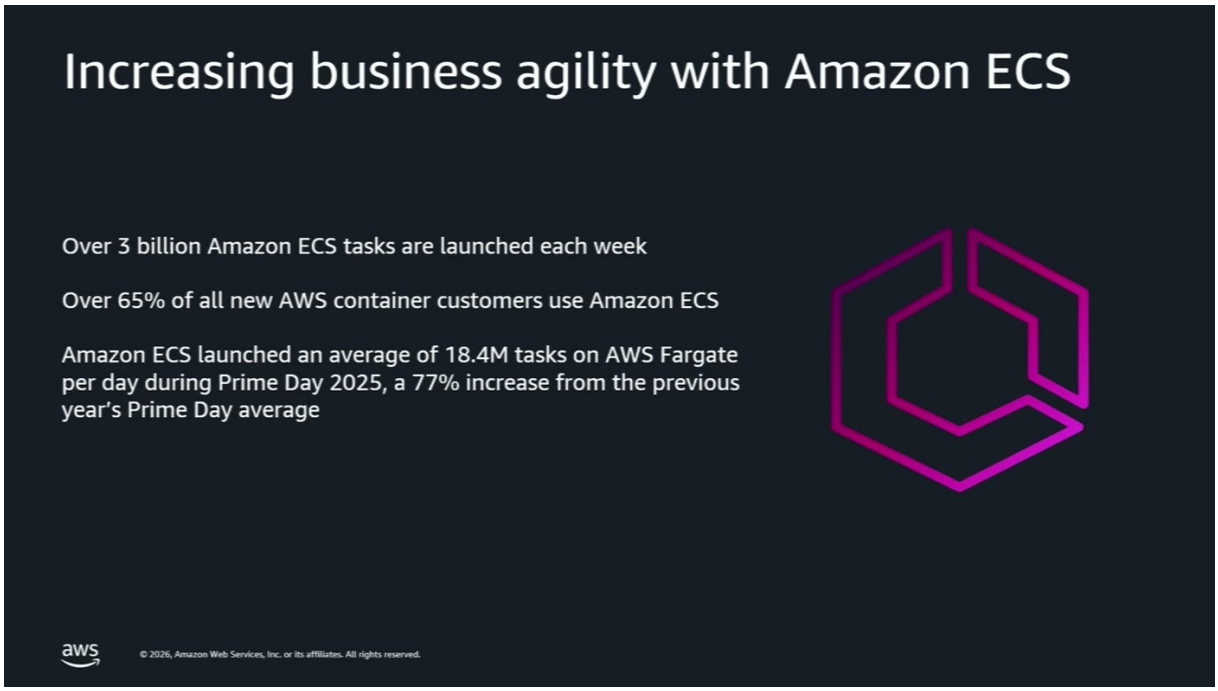


שקופית הפתיחה: שלושת הדוברים ותפקידיהם — שני אנשי AWS ולקוח מ-Earnix.

הסשן היה הסשן היחיד בסאמיט שהוקדש ל-Amazon ECS, והוא נבנה סביב טענה אחת: ECS ממשיך לזוז בכיוון של פחות ניהול תשתית ויותר קוד. Evgeny Grinfeld פתח: "והיום אנחנו נשמח לשתף אתכם כיצד Amazon ECS מתפתח, כדי להפוך את האורכסטרציה של הקונטיינרים לפשוטה וחזקה יותר" [0:00].

- **שני חידושים מהשנה האחרונה עומדים במרכז.** ECS Express Mode מקצר את הדרך מקוד לדיפלוימנט, ו-ECS Managed Instances מגשר בין הפשטות של Fargate לשליטה של EC2. שניהם, בלשון הסיכום של Yaki Ratz, "ששניהם דברים שיצאו בשנה האחרונה" [38:46].
- **Express Mode מצמצם את הקונפיגורציה לשלושה פרמטרים.** במקום Task Definition, Service, Load Balancer, Security Groups וסרטיפיקטים — "אתם צריכים לספק לנו רק שלושה דברים" [4:36]: Container Image, Task Execution Role ו-Infrastructure Role.
- **Managed Instances הם EC2 בחשבון שלכם שמנוהלים במלואם על ידי ECS.** Scaling ברזולוציה של 20 שניות [6:07], מערכת הפעלה Bottlerocket, החלפת אינסטנס תוך 14 יום לכל היותר (ועוד שבוע גרייס), ובלי גישת SSH או root [10:43].
- **הארביטראז' הוא בין בידוד לניצולת.** Fargate מריץ טסק אחד לכל node ולא ממחזר אותו; Managed Instances "מכניסים כמה טסקים לאותו אינסטנס עם BIN Packing חכם" [10:43] — יותר utilization, פחות בידוד.
- **האופטימיזציה מתפרסת על ארבעה צירים.** חומרה (Intel / AMD / Graviton), מודל רכישה (On-Demand, Savings Plans עד 72%, Spot עד 90%), scaling ו-observability [12:17].
- **סיפור הלקוח הוא לא סיפור על ECS אלא על צוואר בקבוק ברשת.** Earnix הפכה את טסקי ה-ECS שלה ל-swarm של BitTorrent בתוך ה-VPC, וג'וב פרודקשן על טבלה של 500 מיליון שורות ירד מטיימאאוט של שלוש שעות לפחות מעשר דקות [37:14].

2. הרקע: למה ECS, ובאילו מספרים



שלושת המספרים שהוצגו לביסוס אימוץ השירות, כולל נתוני Prime Day 2025.

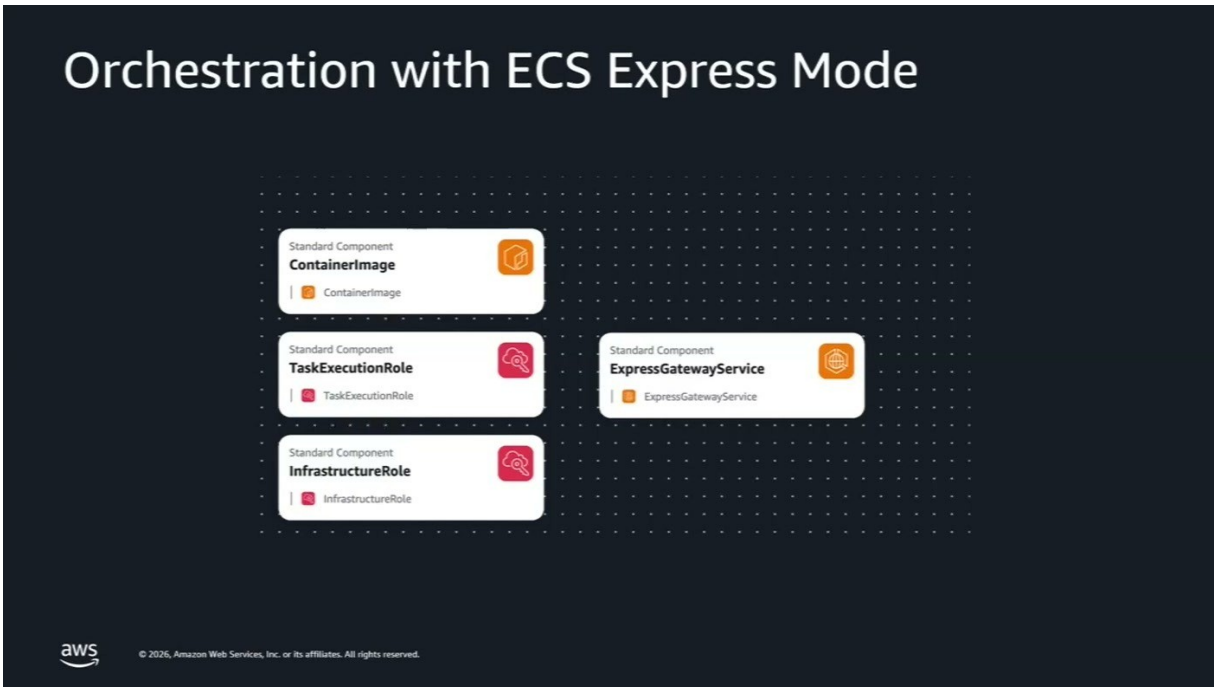
Grinfeld פתח בארבעה יסודות של ECS: הרצת קונטיינרים בסקייל, תשלום לפי שימוש בלבד ("אין תשלום על ECS עצמו"), הסרת ה-operational overhead, והטמעה של AWS Security Best Practices כברירת מחדל [1:33]. המספרים שעל השקופית: מעל 3 מיליארד טסקים בשבוע, מעל 65% מלקוחות הקונטיינרים החדשים של AWS, ובממוצע 18.4 מיליון טסקים ביום על Fargate במהלך Prime Day 2025 — עלייה של 77% לעומת השנה הקודמת.

שלוש הסיבות שהוצגו לאמון הלקוחות: פשטות, אבטחה ויעילות. את היעילות הוא הדגים דרך מודל החיוב של Fargate — "כשאתם מבקשים שני VCPU ושמונה גיגבייט של ממאורי זה בדיוק מה שאתם מקבלים ומשלמים עליו וכל ה-overhead זה כבר בעיה שלנו" [3:03].

3. ECS Express Mode: שלושה פרמטרים במקום עשרות

החלק הזה נפתח בטענה על זמן המפתח: הרצון למקסם זמן על Business Logic ולמזער זמן על הגדרת Security Groups, Load Balancers וניהול סרטיפיקטים. "מדובר בלהביא אתכם מרעיון לקוד ולדיפלוימנט מהר ככל שאפשר בזמן שדבוייץ מטפלת בכל העניין של האינפרסטרקצ'ה" [3:03].

השקופית שקדמה לזה (ECS Landscape) פרשה את כל רכיבי ECS שמפתח נדרש להכיר — Task Definition, Services, Load Balancing, Security Groups ועוד. Express Mode, לדברי Grinfeld, "מפשט את זה דרמטית" [4:36].

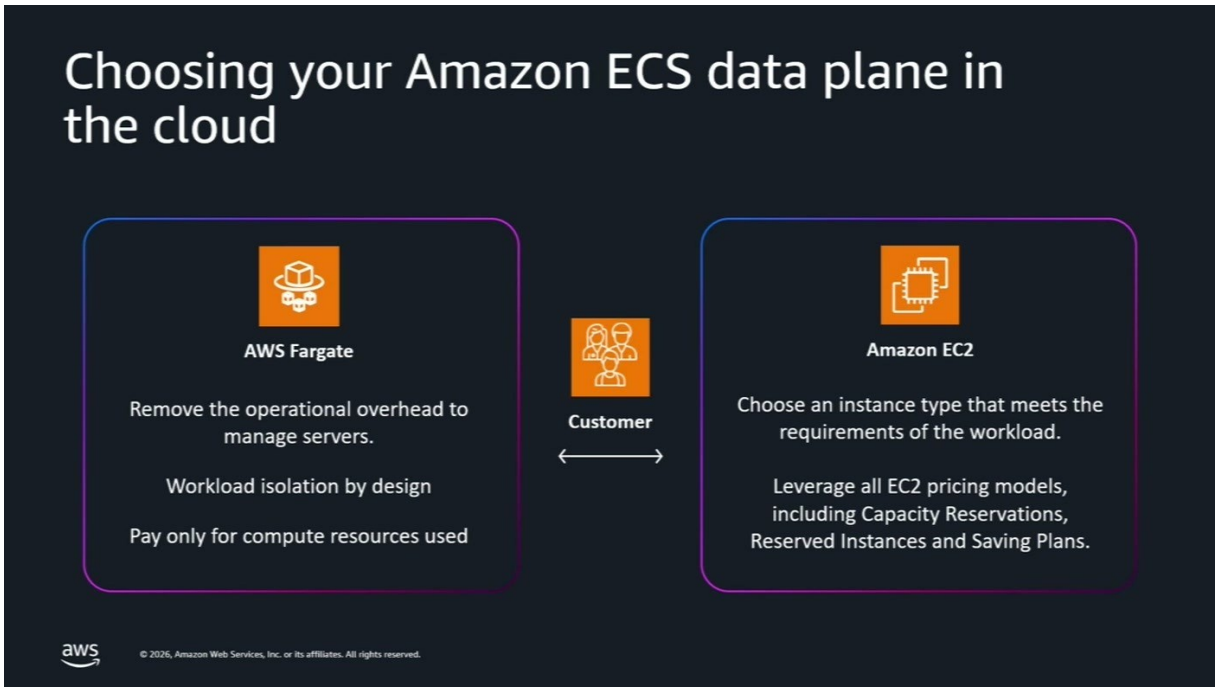


שלושת הרכיבים שהמשתמש מספק ב-Express Mode, מול ה-ExpressGatewayService שנוצר אוטומטית.

- **Container Image —** האפליקציה עצמה.
- **Task Execution Role —** ההרשאות לאפליקציה שלכם.
- **Infrastructure Role —** הרשאות ל-AWS להריץ ריסורסים בשמכם.

Evgeny Grinfeld, [4:36] — זהו, AWS מטפלת בכל שאר. Auto Scaling, Load Balancers, Security Groups, וכל ה-Life Cycle Policy של האפליקציה שלכם.

4. שלוש אפשרויות Capacity



שני קצוות ה-data plane: Fargate מול Amazon EC2, והשיקולים שמפרידים ביניהם.

- **Amazon EC2 (self-managed):** "אתם מנהלים את האינסטנצים בעצמכם, שליטה מליאה על סוגי אינסטנצים ומודל הרכישה, אבל אתם אחראים על פטשינג וקנפיגורציה" [4:36].
- **AWS Fargate:** Serverless מלא. "אין אינפרסטרקש לנהל, אין פאצ'ים לעשות, כל טסק רץ באינסטנס משלו ואתם משלמים רק על זמני ריצה" [6:07]. ברגע שהטסק נעצר — החיוב נעצר.
- **ECS Managed Instances:** האפשרות השלישית, שהוכרזה בשנה שעברה, "בדיוק משיג את האיזון הזה בין פשוטות של Fargate לשליטה של EC2" [6:07].

5. ECS Managed Instances: מה זה נותן

ארבע ההבטחות של ההצעה: חדשנות, ביצועים וזמינות, עלות ואבטחה.

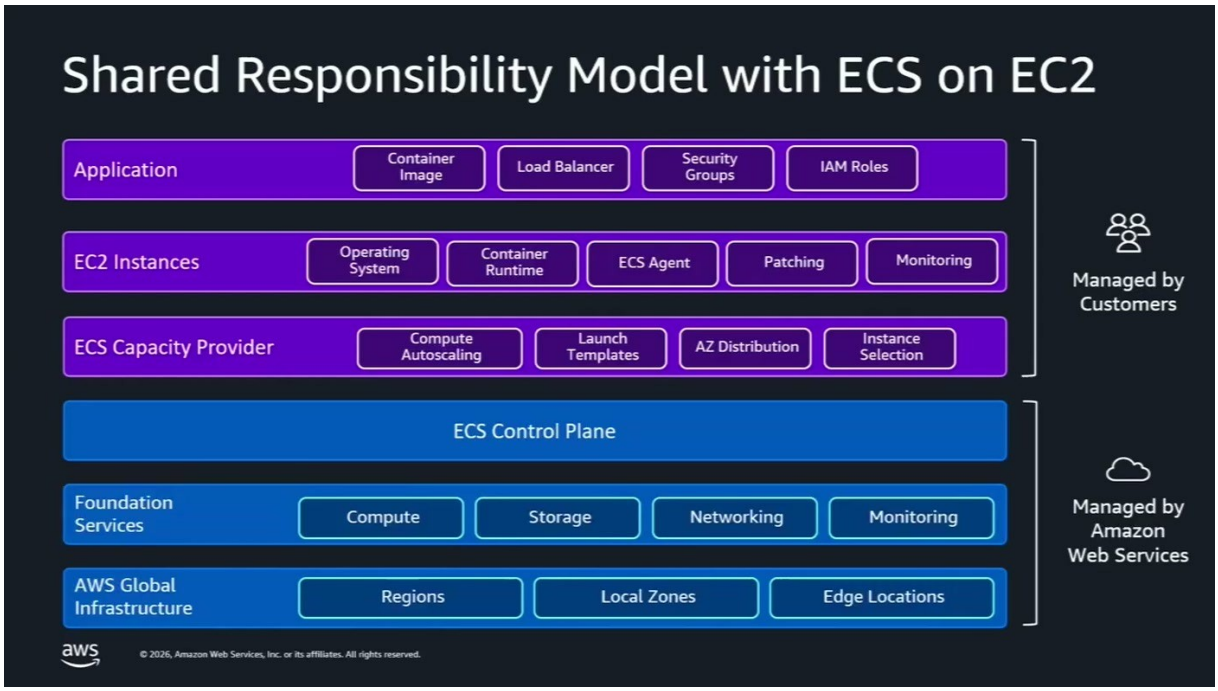
המנגנון: אינסטנסי EC2 רצים בחשבון הלקוח, אבל מנוהלים במלואם על ידי ECS — כולל כל ה-life cycle. שתי נקודות טכניות שנאמרו במפורש: "אנחנו מבצעים scaling ברזולוציה של 20 שניות לתגובה מהירה יותר" [6:07], ומערכת ההפעלה היא Bottlerocket עם סט מינימלי של פקג'ים.

ברירת המחדל היא בחירת אינסטנס לפי עלות, אבל אפשר "להגדיר בדיוק מה נדרש" — Graviton, נטוורקינג ייעודי, GPU, ואף Trainium ו-Inferentia לעומסי AI [7:41].

Managed Daemons

רכיב נפרד שנבנה לתוך Managed Instances: ניהול אוטומטי של סוכני תפעול — CloudWatch agent, Datadog, Fluent Bit — בלי תלות בצוותי הפיתוח. "הדימון עולה לפני אפליקיישן טסק וירד האחרון. ואז אין לכם גב במוניטורינג" [7:41]. עדכוני הסוכנים מתבצעים ב-rolling deployment אוטומטי ללא downtime.

6. מודל האחריות המשותפת ובידוד



חלוקת השכבות בין מה שהלקוח מנהל למה ש-AWS מנהלת, בהרצה על EC2.

"הבנת מודל של shared responsibility היא קריטית" [9:13]. עם אינסטנסים מנוהלים עצמית, AWS אחראית רק ל-ECS control plane ולתשתית היסודית, והלקוח אחראי ל-capacity, scaling, פיזור בין AZ-ים, patching ו-monitoring — "וזה יכול להפוך למורכב למידי ככל שאתם גדלים". עם Fargate ועם Managed Instances הגבול זז: AWS לוקחת את הקפסיטי, ניהול האינסטנסים, האבטחה והפאטצ'ינג, והלקוח נשאר עם האפליקציה, ה-load balancers, ה-security groups וה-IAM roles.

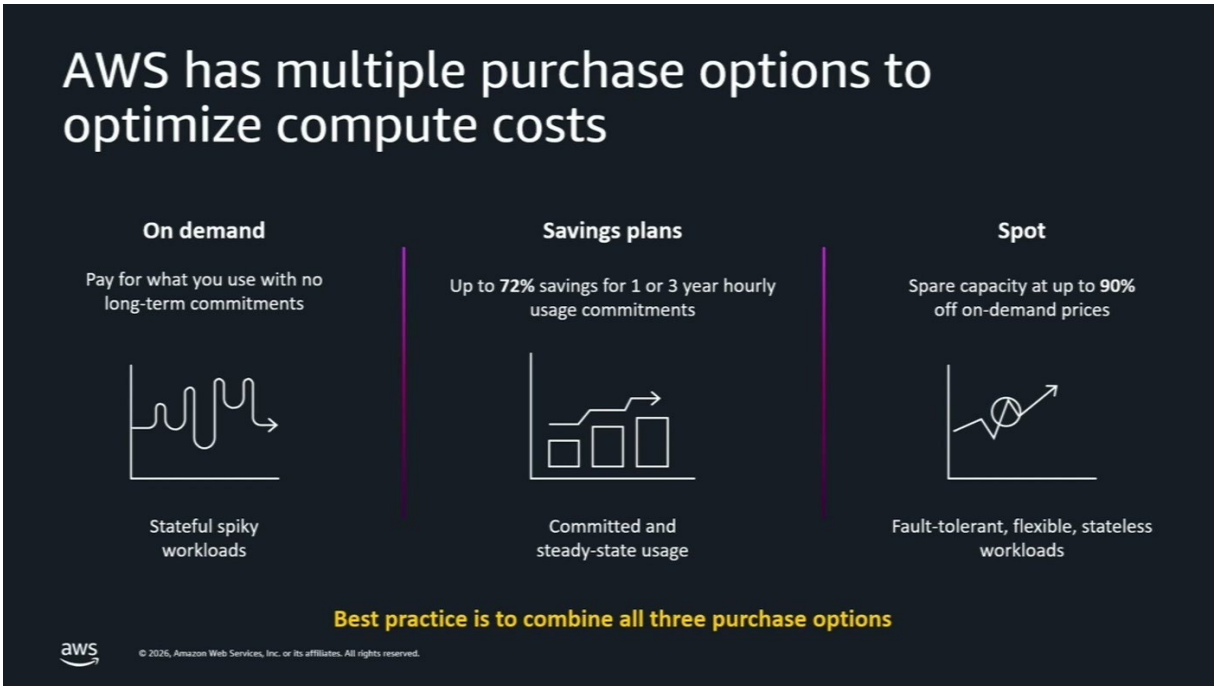
איפה עובר הגבול בין Fargate ל-Managed Instances

ההבדל המשמעותי, לדברי Grinfeld, הוא מודל הבידוד. Fargate נותן את הבידוד החזק ביותר — כל טסק על node משלו, שלא נעשה בו שימוש חוזר. לעומת זאת, ב-Managed Instances "אנחנו מכניסים כמה טסקים לאותו אינסטנס עם BIN Packing חכם שהוא משפר את ה-utilization וכמובן מוריד את העלויות" [10:43]. מי שצריך בכל זאת הפרדה יכול להגדיר כמה capacity providers, או לקבוע instance size שמביא לבידוד ברמת הטסק.

- **Fargate —** בידוד ברמת VM, אין SSH ואין root, פאטצ'ינג כל 14 יום, ו-awsvpc שנותן לכל טסק ENI ו-IP משלו לשליטה fine-grained דרך VPC Flow Logs ו-Security Groups [10:43].
- **Managed Instances —** פאטצ'ינג כל 14 יום, Bottlerocket עם סט פקג'ים מינימלי, "אין גישה לאינסנסים האלה, הכל אך ורק דרך ECS API" [10:43], ואינטגרציה עם ה-maintenance window שאתם מגדירים, "ככה שאתם יכולים לקבוע מתי פאץ' מתבצע" [12:17].

7. אופטימיזציית עלויות

בנקודה הזו עלה Yaki Ratz. הוא מיפה את האופטימיזציה לארבעה אזורים: חומרה, אפשרויות רכישה, scaling ו-observability — "תמיד לדגום את הדברים עלינו להסתכל כל הזמן איך אנחנו משתפרים" [12:17]. בציר החומרה קיימות שלוש משפחות: Intel, AMD ו-Graviton — אבל ב-Fargate "אנחנו לא יכולים באמת לקבוע איזה שרת מנהל לנו את ה-workload" [13:48], ולכן fine-tuning מחייב capacity provider מבוסס EC2.

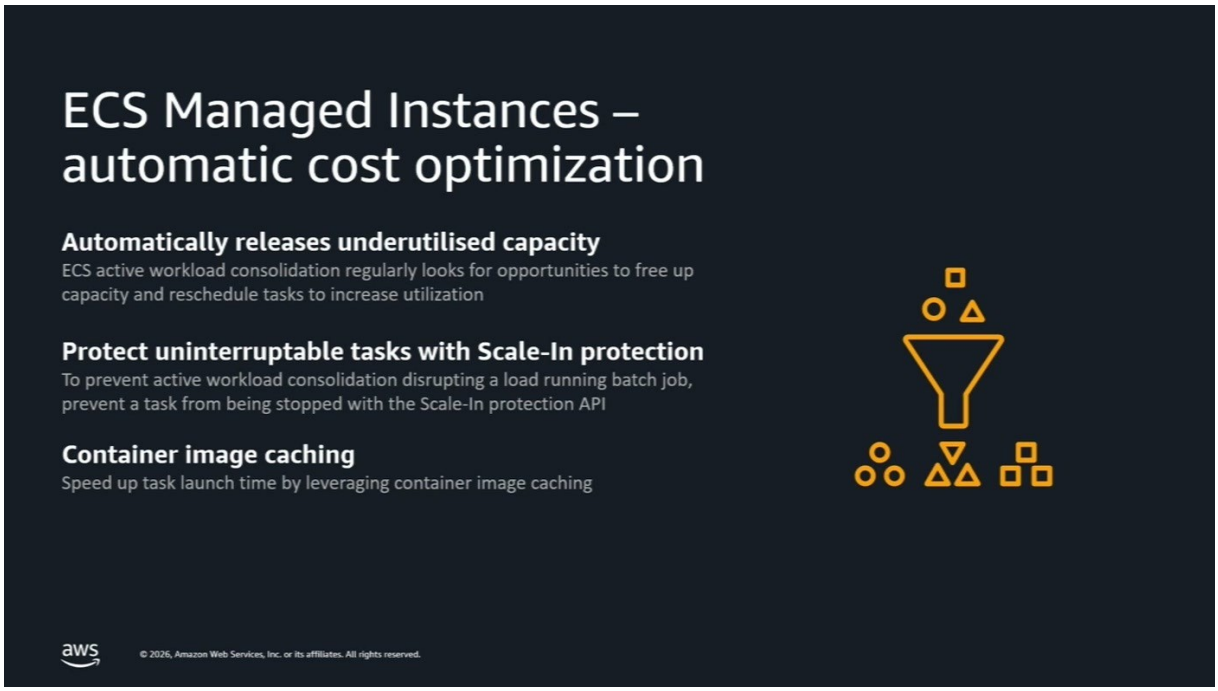


שלושת מודלי הרכישה עם פרופיל העומס המתאים לכל אחד, וההמלצה לשלב את שלושתם.

- **On-Demand —** לעומסים spiky או כאלה שדפוס הפעולה שלהם עדיין לא ידוע [13:48].
- **Savings Plans —** לעומסים מוכרים שרצים 24/7: "ואז אנחנו בצורה כזאת יכולים לקבל הנחה של עד 72% לעומת ON-DEMAND, אם אנחנו מתחייבים לשלוש שנים" [13:48].
- **Spot —** "פה מדובר על ההנחה הגבוהה ביותר של עד 90%, אבל כמובן שהווקלו צריך להיות מתאים לזה" [15:20] — כלומר לרדת תוך שתי דקות מרגע ההתראה.

אופטימיזציה לפי Capacity Provider

ב-Fargate הוצגו שלושה מנופים: pay as you go ("אנחנו לא מנהלים אינסטנסים, אנחנו לא משלמים על אינסטנסים"), Seekable OCI ל-lazy loading של אימג' הקונטיינר שמקצר את זמן העלייה, ו-AWS Compute Optimizer לזיהוי בקשות משאבים שאינן תואמות שימוש בפועל [15:20].



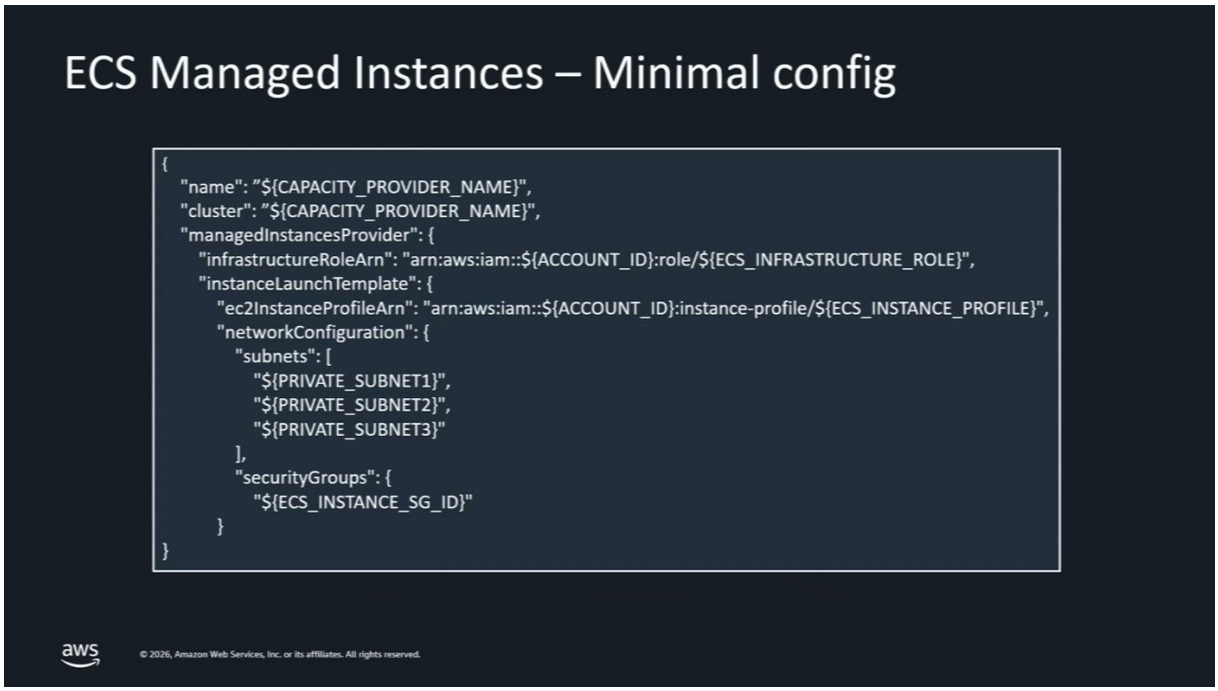
שלושת מנגנוני החיסכון האוטומטיים בצד Managed Instances, כולל Scale-In protection ו-image caching.

- **שחרור אוטומטי של משאבים —** "ה-ECS Managed Instance כל הזמן מסתכל על ה-workload שלכם, ורואה האם אתם רצים על המשאבים העילים עבורכם" [16:51], ומחליף את השרת כשיש אפשרות להתייעל.
- **Scale-In protection —** ההחלפה מכבדת בקשת הגנה של האפליקציה: "אם היא דרשה מאיתנו ברמת הסקיילין API, לחכות את הזמן הדרוש לפני שיורדת, אנחנו כמובן מחכים לזה" [16:51].
- **Container image caching —** האימג' יורד ל-cache של מערכת ההפעלה, ולכן טסק נוסף מאותו סוג על אותו שרת עולה מהר יותר [16:51].

8. Managed Instances מבפנים

8.1 קונפיגורציה מינימלית

"ולכן, בגדול, זה בין שניים-שלושה דברים שאנחנו מגדירים" [18:23]. ה-JSON של ה-capacity provider כולל את הצהרת סוג ה-provider ושלוש הגדרות.

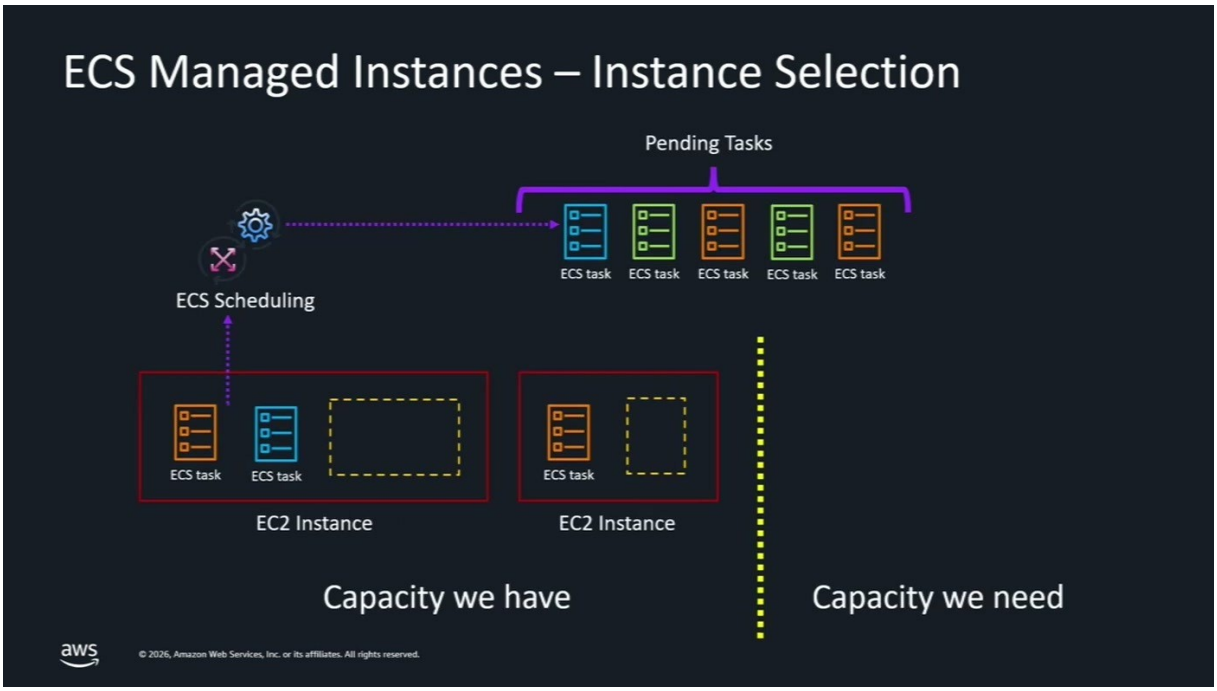


ה-JSON של capacity provider מסוג Managed Instances: infrastructureRoleArn, instance profile, subnets ו-security groups.

- **Infrastructure Role —** הרשאה ל-ECS Managed Instances להקים אינסטנסים בחשבון שלכם.
- **Instance Role —** הרשאה לאפליקציה שרצה על האינסטנס לגשת לשירותי AWS אחרים.
- **Subnets —** ברשימת AZ-ים שונים, וכך "המנג' אינסטנס יפרוס את האפליקציה שלכם, על Multiple Aziz בצורה של Spread Placement" [18:23].

מעבר לזה, זרימת העבודה זהה לעולם Fargate: Task Definition, ECS Service, deploy — "וזהו, אנחנו כבר מוצאים איפה להריץ את זה" [19:56].

8.2 איך נבחר סוג האינסטנס



ה-scheduler משווה בין הקפסיטי הקיימת לקפסיטי הנדרשת עבור ה-pending tasks.

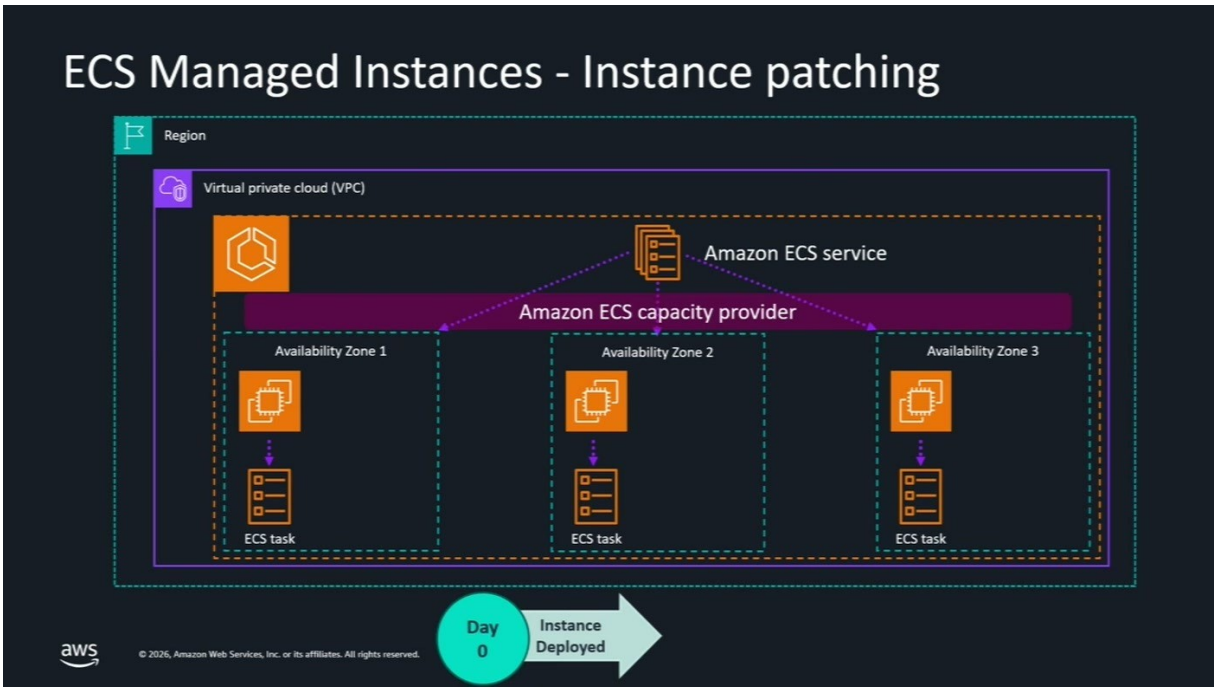
ב-AWS "יש מאות, מעל 900 סוגים" של אינסטנסים [19:56]. ECS Managed Instances נשען על ה-Task Definition — CPU, memory והגדרות נוספות — ומחליט אם יש אינסטנס קיים שמתאים. אם לא, הוא פונה ל-EC2 Fleet API, אותו API שבו משתמש גם Karpenter בעולם EKS, כדי למצוא את השרת שייתן את ה-bin packing הטוב ביותר. ברירת המחדל: "שבי דיפולט, ה-ECS Managed Instance משתמש בשרתים מסוגי C, M ו R" [19:56].

כשצריך משהו אחר נכנס Attribute Based Selection. Ratz הדגים שני מקרים: allow list של שרתי AMD בצד ה-capacity provider יחד עם ארכיטקטורת X86 ב-Task Definition, כדי לקבל את יתרון ה-single core של AMD; ומקרה שני של Inferentia, שבו ה-capacity provider מוגדר לאינפרנציה וה-Task Definition מבקש neuron devices [21:28]. המסקנה שלו: "אפשר מצד אחד או להסתמך על מה ש-ECS מיינגניסטיון יודע לעשות או להשפיע לפי האפליקציה שלנו".

8.3 החלפת שרתים, rebalance ופאטצ'ינג

מכיוון שהאחריות על האינסטנסים עברה ל-AWS, נדרש מנגנון שמחליף אותם באופן שוטף. Ratz מנה ארבע הזדמנויות להחלפה: החלטת האופרטור להוריד אינסטנס, maintenance window שהוגדר, אירועי auto-scaling, ו-cost optimization [23:15]. בתהליך ההחלפה מוקצה שרת חדש, הטסקים המנוהלים על ידי ה-Service מועברים אליו, וטסקים שהורצו ידנית ב-RunTask או StartTask נשארים באחריות הלקוח — "אנחנו סומכים על זה שאתם אחראים על הלייף סייק שלו" [23:15].

בנוסף רץ מנגנון rebalance: אחרי סבב החלפות ייתכן שסרוויס כבר לא מפוזר נכון בין AZ-ים, ואז "ה-ECS Scheduler בודק אם יש דריפט כזה הוא דוגם אותו ומתקן את הבעיה" [24:46] — מעלה טסק חדש ורק אחר כך מוריד את הישן. במקביל, כל stop event משמש הזדמנות לצמצום: שרת שנשארו עליו רק טסקים שנעצרו יורד אוטומטית [24:46].



ציר הזמן של החלפת אינסטנס, מ-Day 0 שבו הוא נפרס.

המדיניות המוצהרת: "אנחנו רוצים לדאוג שהשרתים יוחלפו לכל היותר 14 יום" [26:17]. במשך 14 הימים הראשונים המערכת מחפשת הזדמנות להחלפה; אם לא נמצאה, "יש לנו עוד גרייס של שבוע נוסף" שבו ההחלפה אגרסיבית יותר — אך עדיין מכבדת Scaling Protection. כלומר, גבול עליון של 21 יום לחיי אינסטנס.

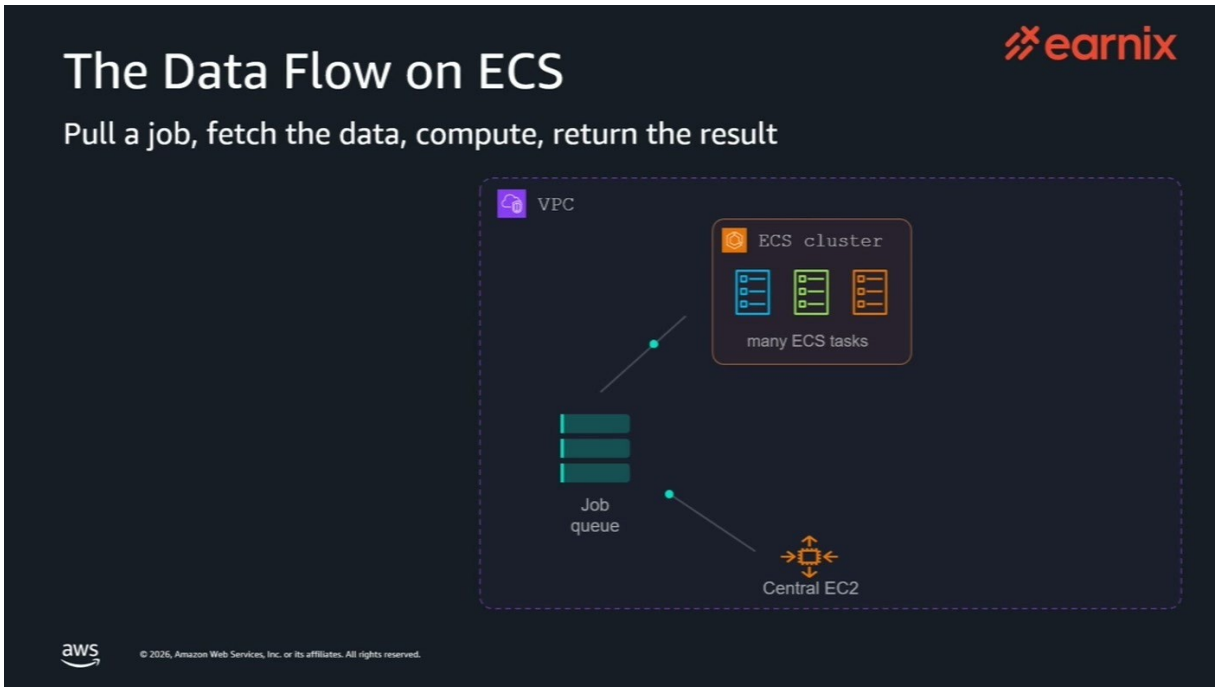
9. מקרה מהשטח: Earnix

Hod Raz, Software Team Leader ב-Earnix, פתח בשאלה לקהל על שימוש ב-BitTorrent, ואז הכריז: "מה אם הייתי אומר לכם שגם בביטוח משתמשים בתורטים" [27:47]. Earnix הוצגה כחברת פינטק גלובלית בת יותר מ-25 שנה, שבונה פלטפורמות AI בזמן אמת לעולמות הביטוח והבנקאות.

המוצר הוותיק, Price-it, מחזיר הצעת ביטוח מותאמת אישית בלחיצת כפתור. מאחורי המחיר האחד עומדים מודלים סטטיסטיים של אקטוארים, אופטימיזציות תחת אילוצי רווחיות ותחרותיות, ורגולציה משתנה בין מדינות — "וכל זה על מערכות שפיתחנו שמחזירות תשובה במילי שניות" [29:21].

9.1 מ-Scale-Up ל-Scale-Out

הארכיטקטורה המקורית הייתה מנוע חישוב על ה-JVM שרץ על אינסטנס EC2 מרכזי אחד. ככל שהחישובים הסתבכו הגדילו את האינסטנס, "אבל באיזשהו שלב, כבר לא יכולנו להגדיל את זה יותר" [30:58]. הפתרון: להשאיר את השרת המרכזי, ולהוציא את החישובים הכבדים ל-scale-out על ECS.



זרימת העבודה: שרת מרכזי מפצל ג'ובים לתור, וטסקי ECS מושכים אותם ומחזירים תוצאות.

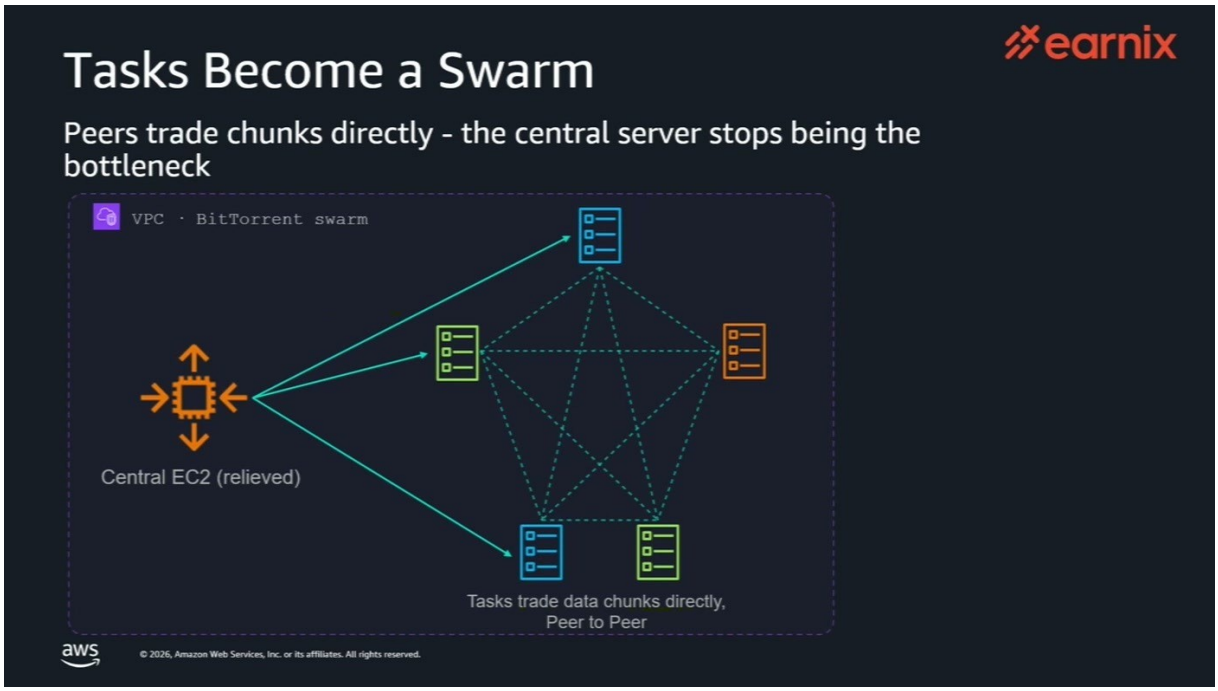
כל טסק עושה ארבעה דברים: מושך עבודה מהתור, מביא את הדאטה מהשרת המרכזי, מחשב, ומחזיר תשובות דרך תור יוצא. מדיניות ה-scale-out נגזרת ממדידה: קודם ריצה על סמפל קטן כדי לאמוד את קצב החישוב, ואז — "אנחנו רוצים שאגב כל טסק ירוץ בסביבות 10 שניות" [32:33] — נקבע גודל הג'וב, מכאן עומק התור, ומכאן מספר הטסקים.

9.2 הבעיה: הטסקים חיכו, לא חישבו

המעבר ל-ECS שיפר דרמטית את זמן ה-CPU, אבל חשף צוואר בקבוק ברשת: כל טסק הוריד את מלוא הדאטה מהשרת המרכזי, וככל שהגדילו את מספר הטסקים החמיר העומס. "רוב הזמן בעצם הטסקים הם לא חישבו הם פשוט חיכו שהדאטה יגיע אליהם" [34:06].

Raz התייחס מראש לשאלה המתבקשת — למה לא S3. התשובה: הדאטה נשמר כהרבה מאוד קבצים קטנים, ולכן עם הרבה טסקים "היינו בעצם מגיעים ל-rate limit" [34:06].

9.3 הפתרון: swarm בתוך ה-VPC



הטסקים מחליפים ביניהם נתחי דאטה ישירות, והשרת המרכזי מפסיק להיות צוואר הבקבוק.

BitTorrent הוצג כפרוטוקול peer-to-peer להפצת קבצים גדולים בסביבה מרובת צרכנים, שמחלק את הדאטה לחתיכות: כל צרכן מוריד חתיכה אחרת ומשתף אותה מיד. "היתרון הגדול שלו הוא שהוא מעלים לחלוטין את צוואר הבקבוק" [35:36] — ובניגוד לארכיטקטורת שרת-לקוח, הוספת קליינטים משפרת את המצב במקום להחמיר אותו.

המימוש: טסקי ה-ECS הפכו ל-swarm בתוך ה-VPC. הצוות השתמש בספריית BitTorrent בקוד פתוח הכתובה כולה ב-Java, ותרם לה יכולת של תיעדוף בהורדה — הטסק מוריד קודם את הנתחים שעליהם הוא צריך לחשב, מתחיל לחשב, ובו זמנית משתף אותם עם האחרים [37:14]. Raz נתן קרדיט למפתח הבכיר Andrew Pickler על הרעיון, על הפיתוח ועל התרומה לקוד הפתוח.

התוצאה על ג'וב פרודקשן אחד: טבלה של 500 מיליון שורות, לפני ואחרי.

> **Hod Raz, [37:14] —** ה-use case ו-production היה טבלה בת 500 מיליון שורות, שהזמן החישוב עליה לא הסתיים גם עד שהוא הגיע לטיימאוט במערכת שלנו, שהוא שלוש שעות. אחרי הפיתוח, זמן הריצה התקצר ל-10 דקות, פחות מ-10 דקות.

האתגר הפתוח שהוצג: "עלייה של כל טסק לוקחת שתי דקות, וזה משהו שאנחנו מנסים לקצר" [37:14]. שני כיווני טיפול — הקטנת האימג', ו-POC שרץ בדיוק עכשיו על Managed Instances, ש"גם יכול להוריד לנו את זמן עליית הטסקים וגם, לא פחות חשוב, להוזיל לנו את המחיר" [38:46].

10. מה לוקחים מכאן

הסיכום של Ratz היה ההמלצה המעשית היחידה שנאמרה במפורש בסשן, והיא מנוסחת לפי נקודת המוצא של המאזין.

- **אם אתם על ECS עם EC2 —** "אנחנו מאוד ממליצים לכם לבחון את ה-ECS Managing Instance מהניסיון שלנו. זה גם מייעל את העבודה שלכם, את האופרציה, וגם משפר את העילות וחוסך לכם כסף" [40:17].
- **אם אתם על Fargate ומרוצים —** "אם אתם משתמשים ב-Fargate ומרוצים, תמשיכו, זה ימשיך להיות אופירינג מוביל ב-[40:17] "ACS.
- **אם אתם על Fargate ונתקלים בקיר —** GPU, או טסקים גדולים מהלימיט של Fargate — זו הנקודה לבחון Managed Instances [40:17].
- **למי שרוצה לקצר** Express Mode **— time-to-first-deploy** הוא המסלול: שלושה פרמטרים, ו-AWS מייצרת את שאר הסביבה.
- **שיקול הבידוד נשאר החלטה ארכיטקטונית —** מעבר מ-Fargate ל-Managed Instances מחליף בידוד ברמת node ב-bin packing. במערכות רגולטוריות זו נקודה שדורשת החלטה מודעת, לא ברירת מחדל.
- **השיעור מ-Earnix אינו על ECS —** אחרי שפותרים את בעיית הקומפיוט, צוואר הבקבוק הבא הוא לרוב I/O. כדאי למדוד כמה מזמן הטסק הוא חישוב וכמה הוא המתנה לדאטה, לפני שמגדילים את מספר הטסקים.

מילון מונחים

מונח	מה זה בהקשר של הסשן
ECS Express Mode	מצב פריסה מפושט: הלקוח מספק image ושני roles, ו-AWS מייצרת את ה-Load Balancer, ה-Security Groups וה-Auto Scaling.
ECS Managed Instances	אינסטנסי EC2 שרצים בחשבון הלקוח אך מנוהלים במלואם על ידי ECS — הקצאה, scaling, patching והחלפה.
Capacity Provider	הישות שמגדירה מאיפה מגיעה הקפסיטי לטסקים: Fargate, ASG, או Managed Instances.
Managed Daemons	סוכני תפעול (CloudWatch, Datadog, Fluent Bit) שמנוהלים בנפרד מהאפליקציה; עולים ראשונים ויורדים אחרונים.
Bin Packing	דחיסת מספר טסקים על אותו אינסטנס להעלאת ניצולת והורדת עלות.
Bottlerocket	מערכת הפעלה מינימלית מבוססת-קונטיינרים המשמשת ב-Managed Instances, עם attack surface מצומצם.
Attribute Based Selection	הגדרת דרישות חומרה (ארכיטקטורה, allow list, accelerators) במקום בחירת סוג אינסטנס מפורש.
EC2 Fleet API	ה-API שבו Managed Instances בוחרים את השרת המתאים לטסקים הממתינים; אותו API שמשמש את Karpenter.
Scale-In Protection	מנגנון שבו האפליקציה מבקשת דחייה של הורדת טסק, ומכובד גם בזמן החלפת אינסטנס.
Seekable OCI (SOCI)	טעינה עצלה של אימג' קונטיינר לקיצור זמן העלייה של הטסק.
Spread Placement	פיזור הטסקים בין Availability Zones, נגזר מרשימת ה-subnets שהוגדרה ב-capacity provider.
BitTorrent swarm	בהקשר של Earnix: קבוצת טסקי ECS שמחליפים ביניהם נתחי דאטה בתוך ה-VPC במקום למשוך הכל מהשרת המרכזי.