רזיליינס ב-Amazon Aurora: תבניות HA ו-DR לסביבה גלובלית

AWS Summit Tel Aviv 2026 ,סיכום סשן — TLV-DAT310

Alex Kronfeld, Senior Solutions Architects, AWS | Shiran Melamed, DevOps Director,-ו Edo Friedman JFrog

10 בספטמבר 2026 | משך: 37 דקות | הופק מהקלטת הסשן

סיכום מאת קובי אלמוג

על המסמך הזה

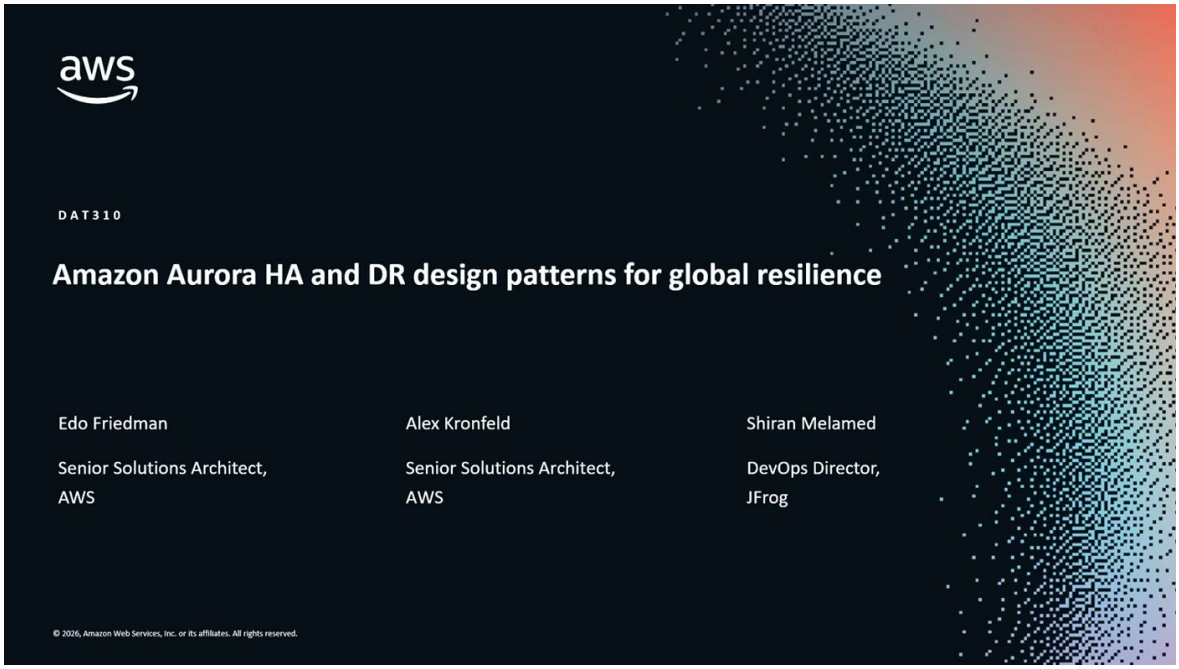
המסמך מסכם את הסשן TLV-DAT310 מתוך AWS Summit Tel Aviv 2026 (מסלול Databases on AWS, רמה 300). הוא נבנה מתמלול אוטומטי של ההקלטה המלאה ומהסליידים שחולצו מהווידאו עצמו, כך שאפשר לחזור לתוכן בלי לצפות שוב ב-37 הדקות. חותמות הזמן בגוף המסמך מפנות לנקודה המדויקת בהקלטה.

הסשן בנוי משני חלקים: חלק תיאורטי של שני ארכיטקטים מ-AWS, שמנוהל כרצף של חמש שאלות — ובכל שאלה מושווית סביבת דאטאבייס בניהול עצמי מול Aurora כשירות מנוהל — וחלק מעשי שבו JFrog מציגה איך התבניות האלה נראות בפרודקשן, ושני אתגרים שהתבניות הסטנדרטיות לא פותרות.

איך לקרוא את זה

- **פרק 1 — תקציר מנהלים:** חמש דקות קריאה. מה הוצג ומה רלוונטי לארגון שמריץ בסיסי נתונים בפרודקשן.
- **פרקים 2–5 — רזיליינס בתוך ריג'ן:** שכבת האחסון, Multi-AZ, סקייל של קריאות, וניהול connections.
- **פרקים 6–9 — חוצה ריג'נים ותחזוקה:** Aurora Global Database, ההבדל בין failover ל-switchover, ושדרוגי גרסה.
- **פרקים 10–13 — המקרה של JFrog:** ארכיטקטורת הקטלוג, טיפול בשחיתות נתונים, והגנת כופר.
- **פרק 14 — מה לוקחים מכאן:** מסקנות ונקודות להמשך.

הערת מקור המסמך הופק מתמלול אוטומטי של ההקלטה. התמלול מדויק בקטעים המקצועיים, אך שמות ומונחים לועזיים מופיעים בו בשיבוש פונטי, והם נורמלו במסמך זה למונח האנגלי המקורי מול הסליידים. הציטוטים נבדקו מול ההקלטה בחותמת הזמן המצוינת ומובאים כלשונם בתמלול, כולל השיבושים. מספרים המופיעים על הסליידים צוינו ככאלה.



שקופית הפתיחה עם שלושת הדוברים ושיוכם הארגוני

1. תקציר מנהלים

הסשן לא מנסה לשכנע שצריך HA ו-DR. הוא לוקח כמובן מאליו שצריך, ושואל שאלה אחרת: כמה מהעבודה הזאת אתם עדיין עושים בעצמכם, וכמה מזה כבר מובנה בשירות. המבנה — חמש שאלות, ובכל אחת "מה קורה בניהול עצמי" מול "מה קורה ב-Aurora" — הוא למעשה טבלת פערים תפעולית.

- **ההפרדה בין compute ל-storage היא מה שמייצר את ה-RPO.** שכבת האחסון של Aurora היא fleet מרובה-דיירים הפרוס על שלושה Availability Zones; כל כתיבה של הדאטאבייס מתורגמת מאחורי הקלעים לשש כתיבות, והדאטאבייס ממתין לחלקן בלבד. התוצאה שהוצגה: "כלומר, ה-RPO הוא 0, קרוב ל-0" [3:04].
- **כמעט כל מספר שהוצג הוא מספר של דקה או פחות.** Failover של replica בתוך ריג'ן — עד 30 שניות [6:10]; switchover חוצה-ריג'נים — RPO אפס ו-RTO עד 30 שניות [15:23]; Blue/Green בתוך ריג'ן — RTO עד 5 שניות [21:28]. הסדר גודל הזה הוא מה שמאפשר את ההבטחה של JFrog ללקוחותיה.
- **ההבחנה המרכזית של החלק השני: failover מול switchover.** Failover הוא מעבר חד-צדדי שעלול לאבד נתונים בגובה ה-replication lag; switchover הוא מעבר מתואם עם RPO אפס. הכלל שניתן מהבמה: "פייל אובר עושים כאשר אין ברירה סוויץ' אובר עושים בשאר המקרים" [16:53].
- **התחזוקה היא הפרק שהכי הופתעתי ממנו.** לא רק שדרוגים מנוהלים, אלא מדיניות ארגונית — AWS Organizations Upgrade Rollout Policy — שמחלקת מאות דאטאבייסים לגלים לפי תגיות סביבה ומשדרגת אותם בהדרגה עם חלון בדיקה בין גל לגל [18:26].
- **JFrog מריצה את זה בשבעה ריג'נים, ומוסיפה שני אתגרים שהתבניות לא מכסות.** קלאסטר primary ב-us-east-1 עם שישה קלאסטרי reader נוספים [27:36]. שני האתגרים שנשארו פתוחים אחרי כל התבניות: שחיתות נתונים שמתפשטת תוך שנייה לכל הריג'נים, ו-ransomware שמפיל את כל החשבון ולא ריג'ן בודד.
- **היכולת הכי חסכונית שהוצגה בדרך אגב: headless cluster.** בקלאסטר משני אפשר להקצות אחסון בלבד, בלי compute — "ואז אתם מקבלים headless קלאסטר עם ריפליקציה מאוד מהירה ברמת הסטוריץ', אבל בלי לשלם על הקומפיוט" [13:51].

רלוונטיות לארגון פיננסי

שלושה דברים במסמך הזה נוגעים ישירות לדרישות רגולטוריות מוכרות: יעדי RTO/RPO מדידים לכל תרחיש כשל, מנגנון מסודר לשדרוגי גרסה בקנה מידה ארגוני עם ראיות לבדיקה בין גלים, והפרק האחרון — הפרדת עותק הגיבוי לחשבון מבודד עם מפתח הצפנה נפרד ו-vault בלתי-משתנה. הפרק האחרון גם מציג בפירוש את הפער שנוצר כאשר הצפנת הדאטה נעשתה במפתח ברירת המחדל של AWS ולא ב-Customer Managed Key.

2. המסגרת: רזיליינס, RTO ו-RPO

הדובר הראשון, ארכיטקט ב-AWS, פתח בהגדרה: "רזיליאנס היא למעשה היכולת האוטומטית להתאושש מתקלה" [0:00], להקצות משאבים באופן דינמי ולענות על ביקוש. הוא פירק אותה לשני מדדים — Availability, הזמינות בתקופה מוגדרת, ו-Disaster Recovery, אוסף הפתרונות הטכנולוגיים להתאוששות מתקלה.

את ה-DR עצמו הוא פירק לשני מדדים נוספים, ואלה שני המספרים שחוזרים לאורך כל הסשן: RTO — Recovery Time Objective, "פרק הזמן שלוקח להתאושש מתקלה", ו-RPO — Recovery Point Objective, "פרק הזמן המקסימלי שבו מתאירים לנו לאבד מידע בזמן תקלה" [1:31].

המסגרת שהוצגה למשך כל החלק התיאורטי: חמש שאלות, ובכל תשובה שתי סביבות — דאטאבייס בניהול עצמי מול Aurora כשירות מנוהל. על כל שקופית סומן בצד ימין לאיזו משתי הסביבות היא שייכת.

3. שאלה 1: מה קורה כשהדאטאבייס נופל

בניהול עצמי

ההנחה: אינסטנס אחד שמחזיק גם את האחסון וגם את המנוע, ומגבה ל-S3. הגיבוי צורך משאבים על חשבון הדאטאבייס ופוגע בביצועיו, ולכן ההחלטה הטבעית היא להריץ אותו פעם ביום — ומכאן ש-"ה-RPO שאתם עלולים לקבל הוא עלול להיות 24 שעות" [1:31–3:04]. ואם האינסטנס עצמו נופל, גם האחסון וגם המנוע אינם זמינים, וההתאוששות עלולה להימשך שעות.

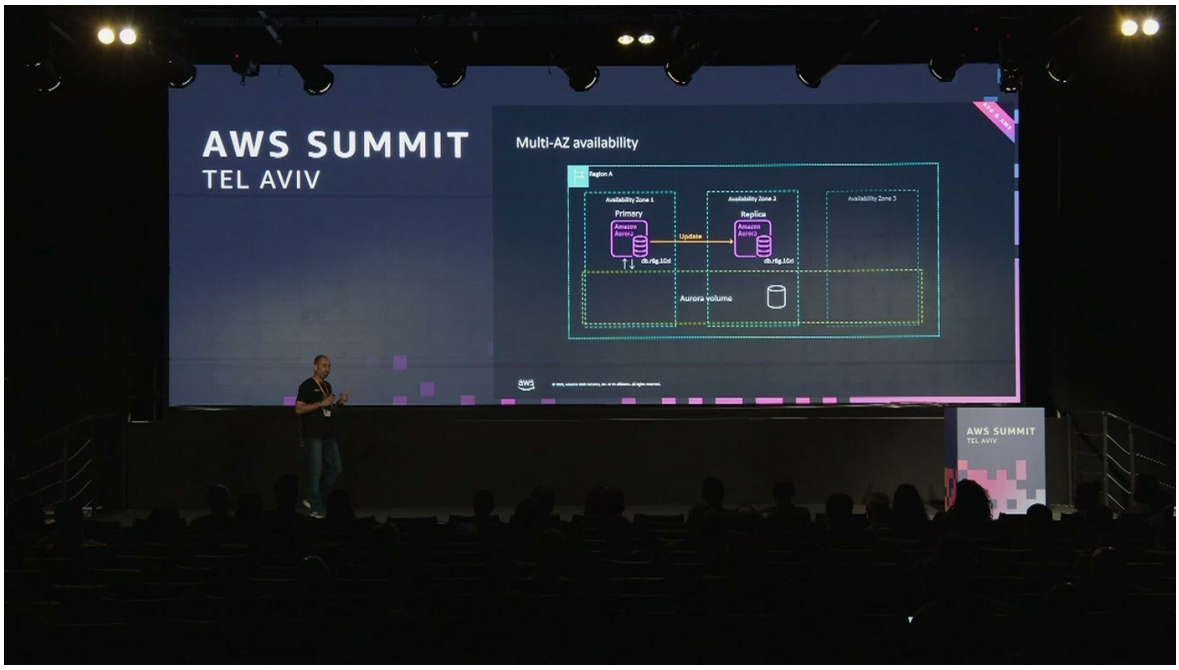
שכבת האחסון של Aurora

ב-Aurora שכבת ה-compute ושכבת ה-storage מופרדות. האחסון הוא fleet מרובה-דיירים של storage nodes הפרוס על שלושה Availability Zones. "בכל כתיבה שהדאטאבייס עושה מאחורי הקלעים מתבצעות למעשה שש כתיבות לשלושה ולאביליטי זונס" [3:04], והדאטאבייס ממתין לאישור של חלקן בלבד. המשמעות: גם אם ייפלו ה-storage nodes של Availability Zone שלם ועוד אחד, הדאטה לא נפגע — "כלומר, ה-RPO הוא 0, קרוב ל-0" [3:04].

מעבר לעמידות, המבנה הזה משנה את אופן הגיבוי. הדאטאבייס מזרים log records לאחסון באופן רציף, ותהליך הגיבוי כותב ל-S3 גם database pages וגם log records ברציפות. שתי נקודות הודגשו: זה לא נעשה על חשבון משאבי הדאטאבייס ולכן לא פוגע בביצועיו, והכתיבה רציפה לאורך ציר הזמן — וזה מה שמאפשר Point-in-Time Recovery.

> **ארכיטקט [4:37], AWS —** חלון הגיבוי הוא הטווח שבין עד לפני חמש דקות לעד לפני 35 ימים תלוי בהגדרות שלכם

בשחזור לנקודת זמן, האחסון משחזר מ-S3 את ה-database pages ואת ה-log records עד לנקודה שנבחרה, ומה שנותר ללקוח הוא להרים דאטאבייס שיעבוד מול השחזור הזה [4:37].



Primary באזור אחד, Replica באזור שני, ומתחתיהם נפח Aurora אחד שמשתרע על שלושת האזורים

שכבת ה-compute

במקרה הבסיסי, אם האינסטנס היחיד נופל אפשר להרים אחר במקומו — ורק כשיסיים את תהליך העלייה, אחרי כמה דקות, יוכל להתחיל לענות לבקשות. ב-Aurora אפשר להקצות מראש אינסטנס מסוג replica, ש-Aurora דואגת לשמור מסונכרן מול האינסטנס הראשי: כל טרנזקציה שנכתבת דרך הראשי זמינה גם לו. במקרה של failover, Aurora מקדמת את ה-replica לתפקיד האינסטנס הראשי — "בדרך כלל זה תהליך שלוקח עד 30 שניות" [6:10], ומאותו רגע הוא מתחיל לענות לבקשות נכנסות.

4. שאלה 2: ביצועים וסקייל

בניהול עצמי, כשהעומס משתנה בצורה קיצונית ולא צפויה במהלך היום, יש שתי אפשרויות מקובלות: scale-up — להרים אינסטנס גדול יותר ולנהל מיגרציה של הדאטה; או לחלק את הדאטה בין מספר אינסטנסים בקלאסטר, ולבנות מנגנון שיודע לפזר שאילתות ולאסוף תשובות בצורה קונסיסטנטית. "שני הפתרונות האלה הם לא זולים ולא פשוטים" [7:42], ובנוסף הם לא נותנים מענה ל-noisy neighbors — תהליכים שצורכים משאבים בצורה קיצונית ועלולים לפגוע בתהליכים קריטיים.

ב-Aurora, אותה replica שהוקמה לצורך זמינות משרתת גם את הביצועים: היא מסונכרנת מול הראשי ומחוברת לאותו אחסון, ולכן אפשר להפנות אליה שאילתות קריאה בלבד ולהוריד מהאינסטנס הראשי את עומס הקריאות. לטיפול ב-noisy neighbors אפשר להרים replica נוספת שתשרת רק אותם, בסביבה מבודדת. "סך הכל אפשר להרים בקלאסטר 15 רפליקות" [7:42].

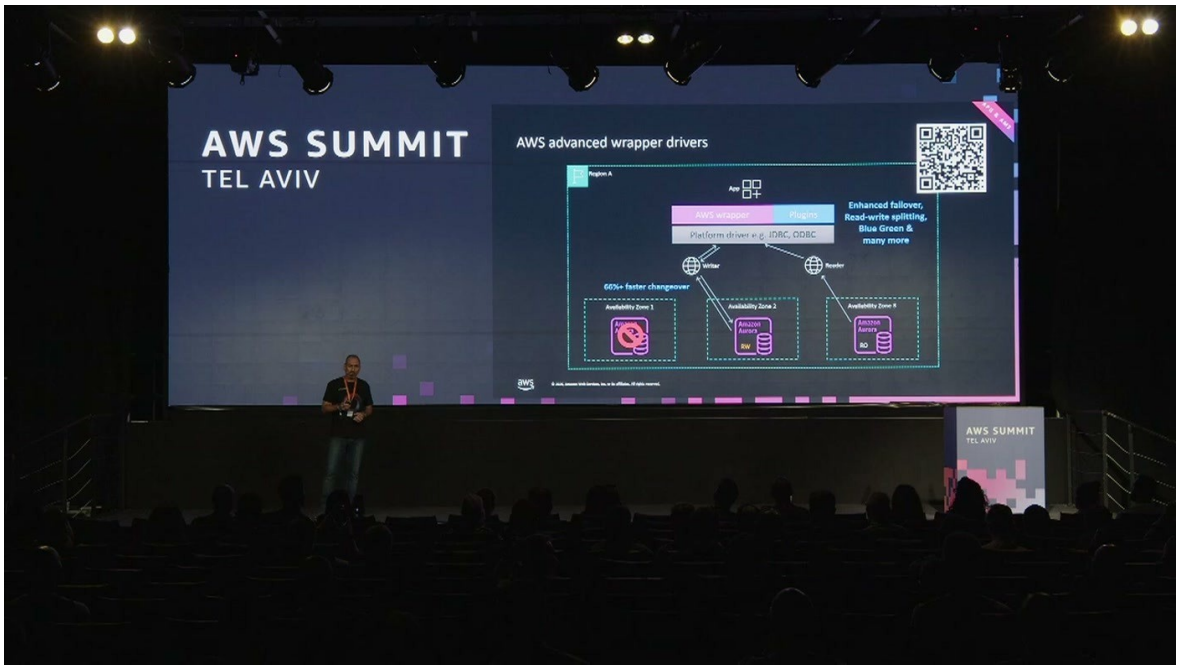
המספרים שהוצגו

- **אחסון:** Aurora מגדילה אותו אוטומטית בהתאם לצריכה, בנפח של עד 256 טרהבייט — בלי פעולה ידנית [7:42].
- **גדלי אינסטנס:** אפשר לשלב replicas בגדלים שונים באותו קלאסטר, עד 192" — 48xlarge קורים" וטרהבייט וחצי זיכרון [9:15].
- **Serverless:** Aurora Serverless מגדיל ומקטין אוטומטית בטווח 0 עד 256 Aurora Capacity Units, כאשר כל יחידה שוות ערך לשני גיגהבייט [9:15].
- **Failover tier:** כששוקלים replicas בגדלים שונים, חשוב שה-replica שתקודם תעמוד בעומס — "הרפליקה שמסומנת במספר הנמוך יותר תתועדף במקרה של failover" [9:15].

5. שאלה 3: ניהול ה-connections

ריבוי connections יקר ופוגע בביצועים, ובקלאסטר נוסף עליו אוברהד תפעולי: צריך לדעת מראש להפנות connection כותב לאינסטנס הכותב ושאילתת קריאה ל-replica. הפתרונות המקובלים בניהול עצמי הם connection pools ו-limits — הגבלת כמות ה-connections לאינסטנס או למשפחת אינסטנסים, וניהול יעיל של מחזור החיים שלהם. "הפיצ'רים האלה קיימים בדרייברים לפוצים כמו Spring ו-JDBC, אבל הם נוטים לחוסר רציבות בעבודה מול קלאסטר" [10:46].

ב-Aurora מוצעים שלושה רבדים. הראשון, RDS Proxy כ-connection pool, שגם מקצר את זמן המעבר במקרה של failover. השני, endpoints מובנים: writer endpoint שמצביע תמיד לאינסטנס הכותב ו-reader endpoint שמצביע ל-replicas שבקלאסטר; במקרה של failover Aurora משנה את ההצבעה של ה-writer, והאינסטנס הישן יורד מ-pool ה-readers [10:46].

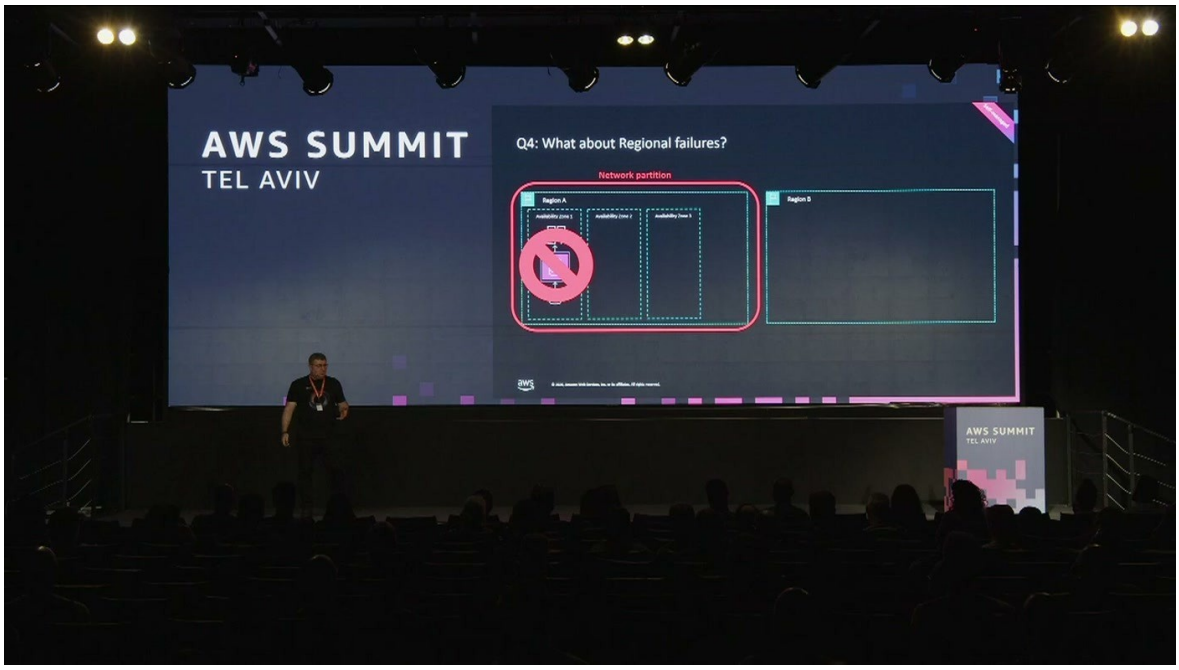


שכבת ה-wrapper מעל דרייבר הפלטפורמה, ולצידה יעד ה-changeover: 66%+ מהיר יותר

הרובד השלישי הוא AWS Advanced Wrapper Drivers — ספריית קוד פתוח שעוטפת דרייברים נפוצים ומרחיבה את יכולותיהם: תמיכה מתקדמת ב-failover, הפרדה בין אינסטנסים קוראים לכותבים, ותמיכה ב-Blue/Green. "השימוש בה יכול לקצר בלמעלה 60%" [10:46] את זמן המעבר ב-failover. על השקופית הופיע QR להורדה, והדובר המליץ עליה במפורש [12:18].

6. שאלה 4: כשל ריג'יונלי ו-Aurora Global Database

כאן החליף את הבמה הדובר השני, שהציג את עצמו: "שמי אלקס קרונפילט ואני Senior Solution Architect ב-AWS" [12:18]. השאלה שלו: מה קורה כשהבעיה אינה ב-Availability Zone אלא ריג'יונלית, או אפילו תקלת תקשורת בין ריג'נים.



התרחיש שמניע את הפרק — network partition שמנתק ריג'ן שלם

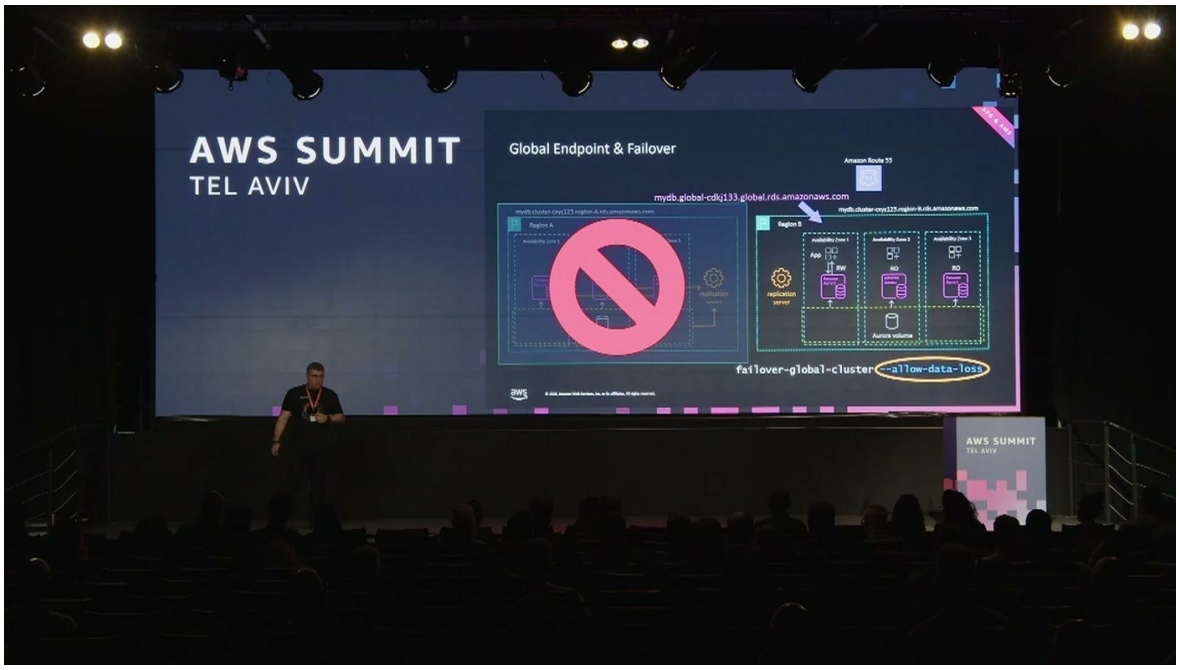
בניהול עצמי, מה שהיינו רוצים הוא עותק של הדאטאבייס בריג'ן אחר עם רפליקציה לוגית ביניהם — "שזה בעצם פעולה לא טריוויאלית ותחזוקה והכל" [12:18]. התשובה של Aurora היא Aurora Global Database: "זאת יכולת לייצר עד 15 סקונדרי קלאסטרס ברידג'נים שונים" [12:18]. כשמייצרים Aurora, secondary cluster מקימה מאחורי הקלעים replication server ו-replication agent, והדאטה משוכפל אסינכרונית מה-primary אל ה-secondary. "RPO הוא עד שנייה אחת" [12:18].

> **סייג** ה-replication lag תלוי במרחק הפיזי בין הריג'נים — "בכל זאת, מהירות האור, אנחנו לא יכולים לשלוט בה" [13:51]. שני ריג'נים בארצות הברית ייתנו ערך נמוך יותר; ריג'ן בארצות הברית מול ריג'ן באוסטרליה ייתן lag גבוה יותר — אך לפי מה שנאמר מהבמה, עדיין מתחת לשנייה.

בקלאסטר המשני אפשר להקים בין 0 ל-15 read-only replicas. הדובר הדגיש שהתמונה שצייר סימטרית רק לצורך ההמחשה: אפשר להקצות אחסון בלבד, בלי compute ובלי read-only replicas כלל — "ואז אתם מקבלים headless קלאסטר עם ריפליקציה מאוד מהירה ברמת הסטוריץ', אבל בלי לשלם על הקומפיוט" [13:51]. זו נקודת חיסכון מעשית עבור מי שמחזיק ריג'ן DR שאינו משרת תעבורה בשגרה.

7. Failover מול Switchover

לפני שתי השיטות הוצגה תשתית משותפת: global endpoint, שמאפשר ללקוחות לא לדעת היכן נמצא ה-primary cluster. הקליינט מתחבר תמיד ל-global endpoint, והוא יודע להצביע אל ה-primary בכל רגע נתון [13:51].



המעבר לריג'ן B, ומתחתיו הדגל שממסן את המחיר: --allow-data-loss

Failover — פעולה שאתם יוזמים

במקרה של אירוע ריג'יונלי, אתם מעבירים באופן פתאומי את ה-primary cluster לריג'ן אחר; אם יש כמה ריג'נים משניים, אתם קובעים מראש את סדר העדיפויות לירושת התפקיד. "אבל תשימו לב להלאו דאטה לוס. במקרה של פיילור, אתם עלולים לאבד דאטה" [15:23] — בדיוק בגובה אותו replication lag של עד שנייה. הדגל שמופיע על השקופית, --allow-data-loss, הוא הביטוי המפורש של המחיר הזה. בסיום המעבר, ה-global endpoint מצביע אל ה-primary החדש בריג'ן ה-DR.

Switchover — פעולה מתואמת

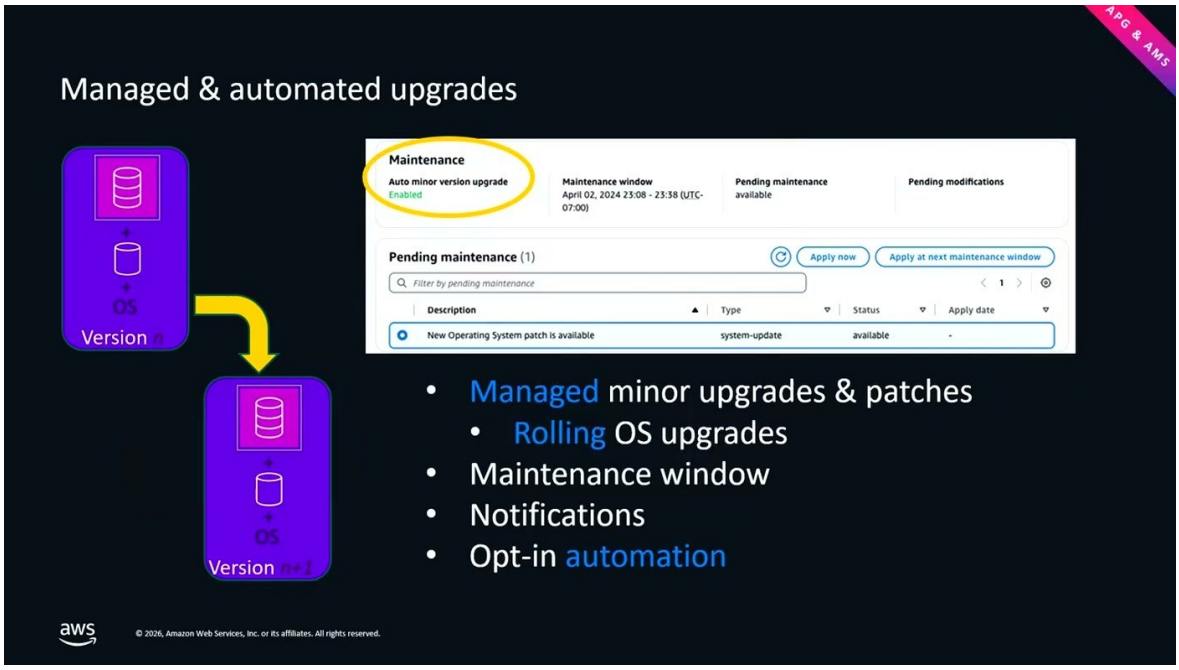
ב-switchover מועבר marker מיוחד בתוך ה-transaction log אל הקלאסטר המשני; ברגע שהוא רואה אותו, הוא נוטל על עצמו את תפקיד ה-primary החדש. זו פעולה מתואמת ומתוזמנת בין שני הקלאסטרים, וההבדל שהודגש כחשוב ביותר: "אתם מקבלים RPO אפס וRTO עד 30 שניות" [15:23].

> **— ארכיטקט AWS, בתשובה לשאלה חוזרת של לקוחות, [16:53]** פייל אובר עושים כאשר אין ברירה סוויץ' אובר עושים בשאר המקרים

כדי להמחיש, הוא הזכיר מהבמה את אירוע 20 באוקטובר 2025 — "אירוע גדול בנורס וירג'יניה שגרם לדאונטיימא מאוד רציני לקומפיוט" [16:53], שנמשך שעות רבות. הנקודה שלו: במהלך האירוע הדאטאבייסים כן הצליחו לשכפל נתונים לריג'נים אחרים, ולקוחות שהחליטו להעביר את ה-workloads שלהם "כן הצליחו לעשות switch over בלי לאבד נתונים" [16:53].

8. שאלה 5: תחזוקה ושדרוגי גרסה

מה בעצם כרוך בשדרוג גרסה של דאטאבייס, למשל מ-MySQL 8.4.1 ל-8.4.8? הרשימה שהוצגה: ביצוע בחלון תחזוקה ("אם אתם לא קובעים את החלון הזה בעצמכם, אנחנו נגבע אותו בשבילכם, אבל חייבים לקבוע" [16:53]), יכולת rollback אם הגרסה החדשה לא מתאימה, תיאום עם הקליינטים והודעה על downtime, בדיקות תאימות ברמת הדאטאבייס וברמת הקליינטים, בדיקות ובחינת ביצועים. "מי שעשה את זה פעם הוא יודע ששדרוג גרסה בדאטה-בס אפילו גרסה משנית זה פעולה מורכבת" [16:53–18:26].

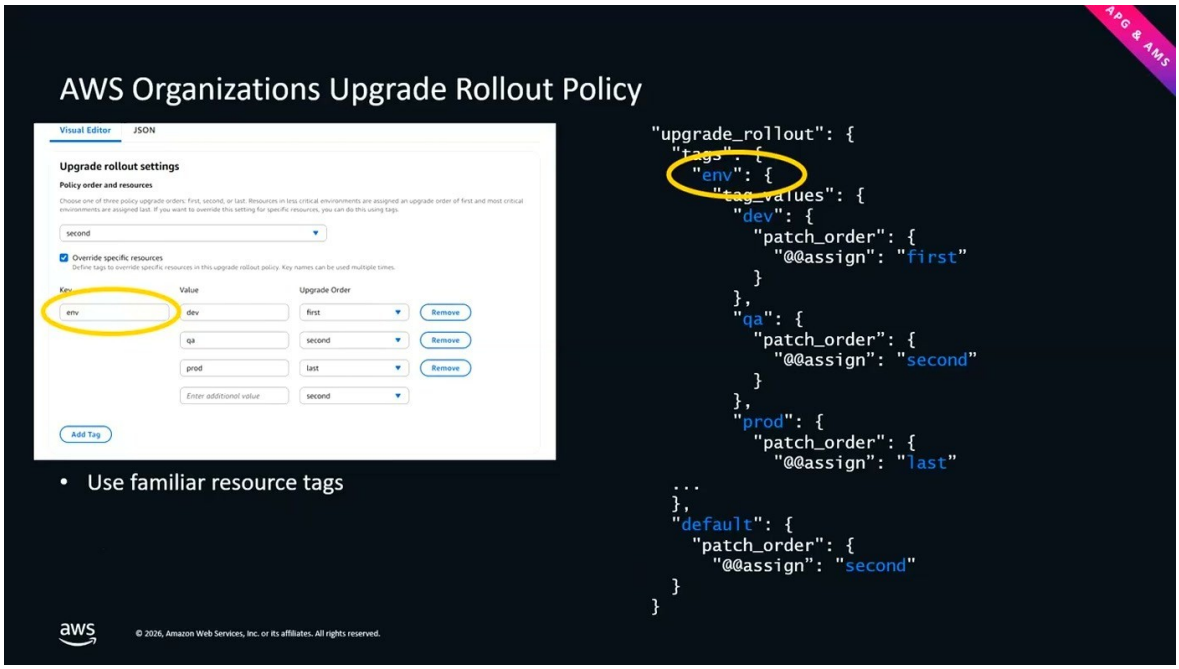


מסך ה-Pending maintenance בקונסולה, ולצידו ארבעת מרכיבי היכולת

התשובה הראשונה היא Managed Automated Upgrades: "אנחנו מכסים שדרוג של גרסות משניות ופטשים" [18:26], וכן עדכוני מערכת הפעלה של האינסטנסים, בחלון תחזוקה ועם נוטיפיקציה מסודרת. הפיצ'ר אופציונלי, אך הומלץ במפורש — הוא מקל על המאמץ התפעולי של הצוותים.

שדרוג בקנה מידה ארגוני

ארגון אנטרפרייז מחזיק המון דאטאבייסים — פיתוח, בדיקות, פרודקשן. לשם כך הוצגה יכולת ברמת הארגון: "יש לנו AWS Organizations Upgrade Rollout Policy" [18:26] — חלוקת הדאטאבייסים לגלים, בדרך כלל לפי תגיות הסביבה. אזהרה שנאמרה בדרך אגב: מי שמחזיק כמות גדולה של דאטאבייסים כדאי שיתייג אותם.

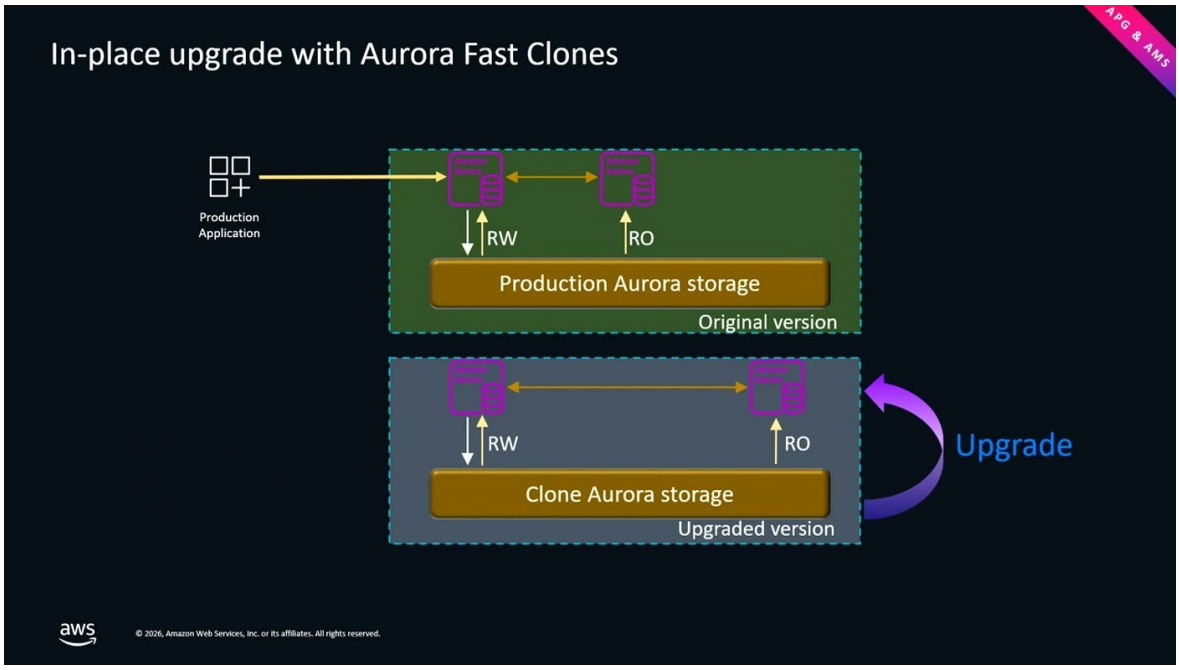


קובץ המדיניות: patch_order לכל תגית, ו-default לדאטאבייסים שלא שויכו לאף גל

הדוגמה שניתנה: development כגל ראשון, בדיקות כגל שני, production כגל שלישי, ודאטאבייסים שאינם נכנסים לאף גל משויכים לגל השני. התהליך: נוטיפיקציה מקדימה על שדרוג מסיבי, שדרוג הגל הראשון, זמן לבדיקה, גל שני, בדיקה, גל שלישי ובדיקה. "זאת שיטה מאוד מסודרת שעוזרת לארגונים גדולים לשדרי כמויות גדולות של דאטאבייסים" [19:57].

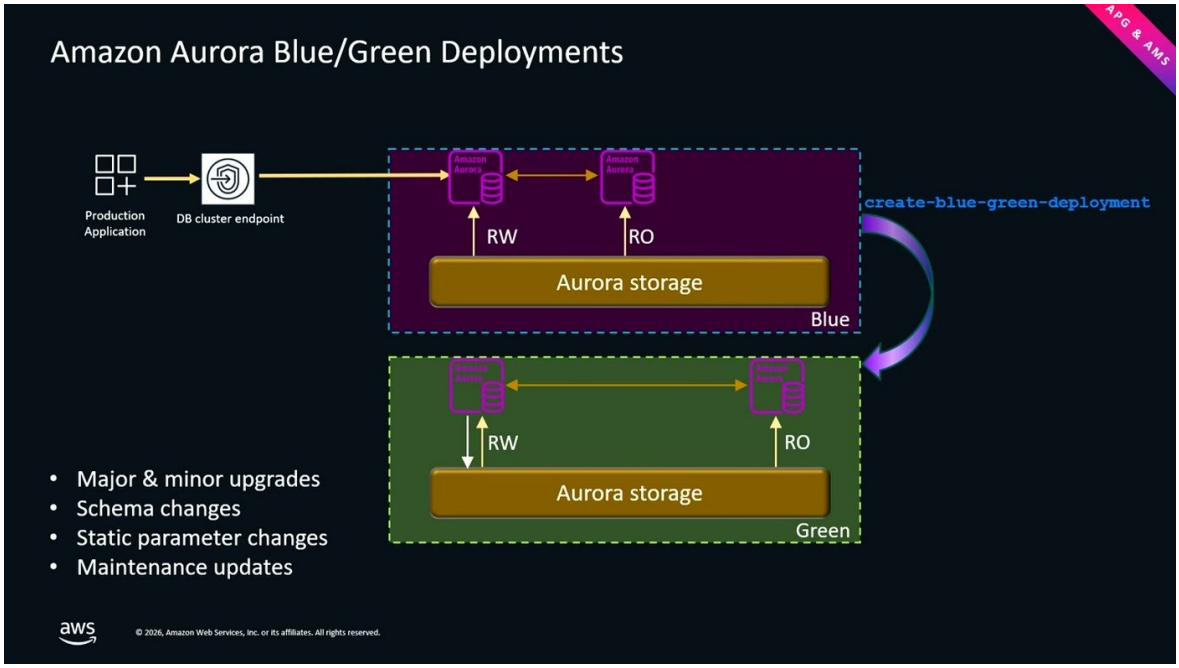
9. Blue/Green Deployments ו-Fast Cloning

האפשרות הראשונה לשדרוג בטוח היא שכפול מהיר: "אורורה תומכת ביצירת קלון מהיר, זה עניין של דקות" [19:57]. על הקלון אפשר להריץ את השדרוג ואת כל הבדיקות — תאימות, ביצועים — ולוודא שהגרסה החדשה תקינה. לאחר מכן זורקים את הקלון ומבצעים את הפעולה על הקלאסטר הראשי.



הקלון מצביע לאחסון המקורי; השדרוג מתבצע על העותק ולא על הפרודקשן

החיסרון הודגש מיד: "מה החיסרון של השיטה הזאת? אורך של דאונטיימא" [19:57] — כל זמן שמריצים את הפעולות על הקלאסטר הראשי, הוא אינו זמין. מכאן השאלה שמובילה לפרק הבא: האם יש דרך לקצר את ה-downtime של ה-primary cluster.



קלאסטר כחול וקלאסטר ירוק מקבילים, ורשימת סוגי השינויים שהמנגנון מכסה

Aurora Blue/Green Deployments מייצר סביבה מקבילה לקלאסטר הראשי. השיטה מכסה את כל סוגי השינויים — גרסאות ראשיות ומשניות, שינוי סכמה, patches ועדכוני תחזוקה. הקלאסטר הראשי הוא "הכחול", המקביל הוא

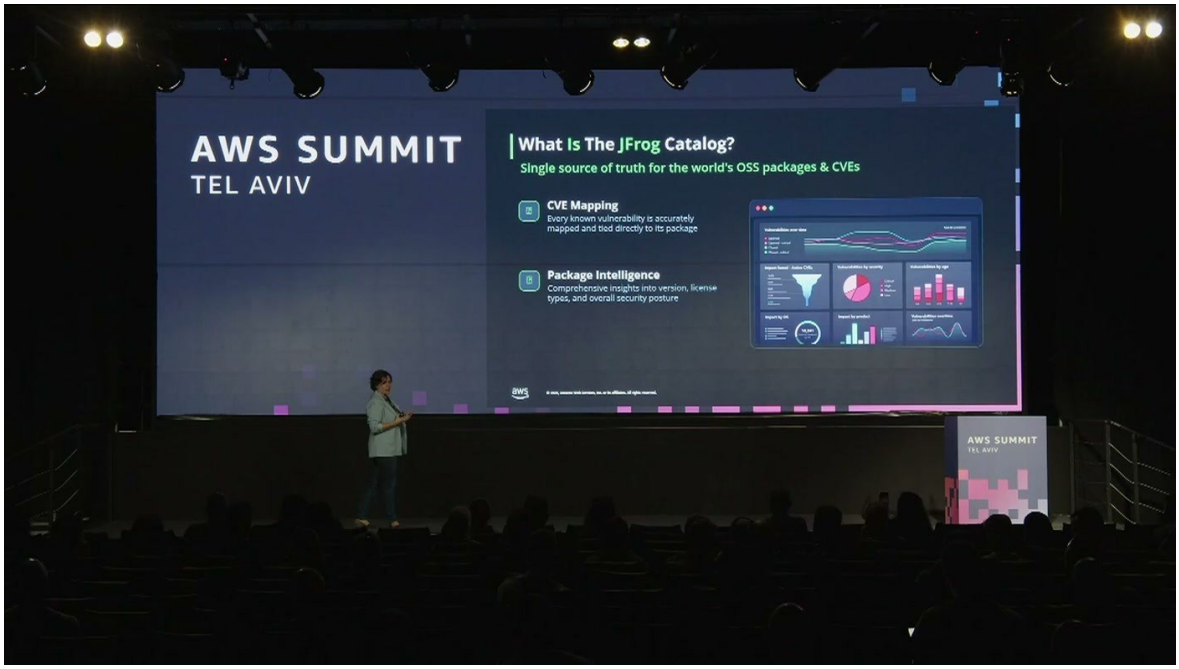
"הירוק", ומתבצעת logical replication מתמשכת מהכחול לירוק כדי שהירוק יישאר מעודכן. מריצים את השדרוג ואת כל הבדיקות על הירוק, וכשמחליטים שהוא תקין — מעבירים את התעבורה.

- **קלאסטר עצמאי בתוך ריג'ן:** "RPO 0, יש logical replication כל הזמן מקלסטר הכחול לקלסטר הירוק, ובתוך הריג'ן RTO עד 5 שניות" [21:28].
- **Global Database:** גם הוא נתמך ב-Blue/Green; ה-RPO נשאר אפס, ו-"רטאו כאן הוא בערך מתחת לדקה" [23:00].
- **שיפור חוויית הקליינט:** עם RDS Proxy לפני הקלאסטר, הקליינטים מתחברים לפרוקסי ובמקום disconnect מקבלים "קביצה ב-latency יחסית נמוכה" [21:28–23:00].

10. המקרה של JFrog: מהו הקטלוג

החלק השלישי הועבר ל-DevOps Director ב-JFrog, שהציגה את עצמה כמי שאחראית לכך "שסביבת הפרודקשן של J-Fog תהיה יציבה, תגדל איתנו, והכי חשוב תהיה משעממת ללא תקלות" [24:32].

JFrog נוסדה ב-2008 סביב Artifactory, והיום היא פלטפורמה שכוללת גם JFrog Security, Curation ו-JFrog UpTrust — האחרון מספק את ההוכחות הנדרשות לצוותי Legal ו-GRC לגבי המוצר. הפלטפורמה מלווה את תהליך הפיתוח משלב ה-commit ועד ההרצה בפרודקשן.



שני התפקידים של הקטלוג: מיפוי CVE ו-package intelligence

הרכיב שעליו נסוב כל הפרק הוא JFrog Catalog: "J-Fog Catalog הוא ה-single source of trust שלנו לכל ה-CVEs שקיימים בעולם" [24:32]. הוא ממפה חבילות וגרסאות ל-CVEs, מספק ראייה של 360 מעלות על החבילה — רישוי, security exposure — ומועשר בדאטה ייחודי של חוקרי JFrog.

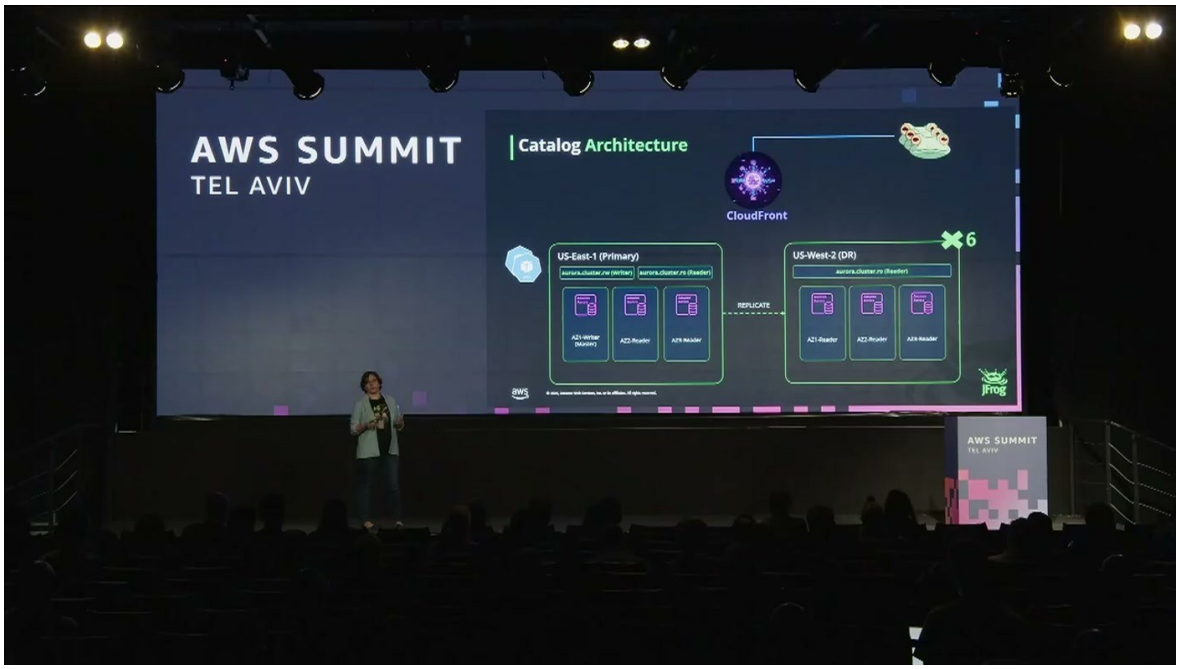
מה נשען על הקטלוג

- **JFrog Curation:** "JFog Curation הוא בעצם Firewall לחבילות לתוך הארגון" [26:02] — חבילה עם CVE נחסמת מלהיכנס, כדי שלא ייווצר סיכון שמישהו ישתמש בה בטעות.
- **JFrog Advanced Security:** נותן תמונת מצב על ה-CVE — כמה הוא נפוץ וכמה הוא בכלל רלוונטי [26:02].
- **JFrog AI Catalog:** מקביל רעיוני ל-Curation, לפיקוח וניהול של מודלי AI [26:02].

למה הקטלוג התאים ל-Aurora

שלוש דרישות הוצגו. הראשונה, הפצה מהירה: "אני מכניסה עדכון לאיזשהו CV חדש אני חייבת שהוא יגיע ללקוחות שלי תוך שנייה" [26:02] — אחרת הלקוח נותר בסיכון. השנייה, דאטאבייס עצום שצריך ניהול אחסון, ואחסון Aurora מנוהל. השלישית, "Fast Recovery, RTO של פחות מדקה" [26:02] לכל אחד מהתרחישים.

11. ארכיטקטורת הקטלוג ותרחישי הכשל



us-east-1 כ-primary עם writer ושני readers, ומולו קלאסטרי reader בריג'נים נוספים מאחורי CloudFront

בצד שמאל ה-primary cluster, שבו יושב ה-writer עם ה-global writer endpoint שאליו מתחברים ה-pods. "והקלאסטר הזה מרופלק לעוד שישה קלאסטרים נוספים שהם הרידר קלאסטרס" [27:36], שאליהם פונים הקליינטים. לקוח של JFrog מגיע ל-CloudFront ומנותב לפי latency ו-geolocation למקום הקרוב ביותר אליו.

ארבעה תרחישים

- **נפילת AZ בקלאסטר reader:** "מה קורה? שום דבר" [27:36]. ה-reader endpoint ממשיך לתפקד, שני אזורים נותרו פעילים, והשירות נמשך.
- **נפילת ה-writer node:** כאן נכנס ה-failover priority שהוגדר מראש — איזה node ראשון לרשת את תפקיד ה-writer (0, 1, 2 על השקופית). "והכי חשוב שהדבר זה נעשה בפחות מדקה" [27:36].
- **נפילת ריג'ן של readers:** ה-pods מזהים דרך liveness ו-readiness שהמערכת אינה זמינה, והלקוחות מנותבים לאחד הריג'נים האחרים — יש רפליקות רבות, וגם ה-writer משרת קריאות [27:36–29:06].
- **נפילת ריג'ן ה-writer:** לקוחות קוראים מנותבים לריג'נים אחרים, ואת ה-writer צריך להעביר ידנית לריג'ן נבחר. ה-writer endpoint נשאר גלובלי ולכן אינו מהווה בעיה [29:06].

מדיניות בפועל בבחירה בין שתי שיטות המעבר, JFrog מיישרת קו עם ההמלצה של AWS: switchover הוא ברירת המחדל משום שהוא מייצר פחות אובדן נתונים, ו-failover נשמר למקרה שאין ברירה [29:06].

12. אתגר א': שחיתות נתונים שמתפשטת

האתגר הראשון שאינו מכוסה על ידי אף אחת מהתבניות: באג באפליקציה או חוקר שהכניס דאטה שגוי. המהירות שהיא יתרון הופכת כאן לחיסרון — "כמו שאמרתי, תוך שנייה, הדאטה השגוע הזה מגיע לכל הריג'נים, לכל הלקוחות שלנו" [29:06]. ההשלכה: אפשר להכניס בתום לב CVE שגוי לסביבת הלקוח, או למנוע ממנו להוריד חבילות תקינות.

ארבע אפשרויות טיפול הוצגו, בסדר עולה של איכות:

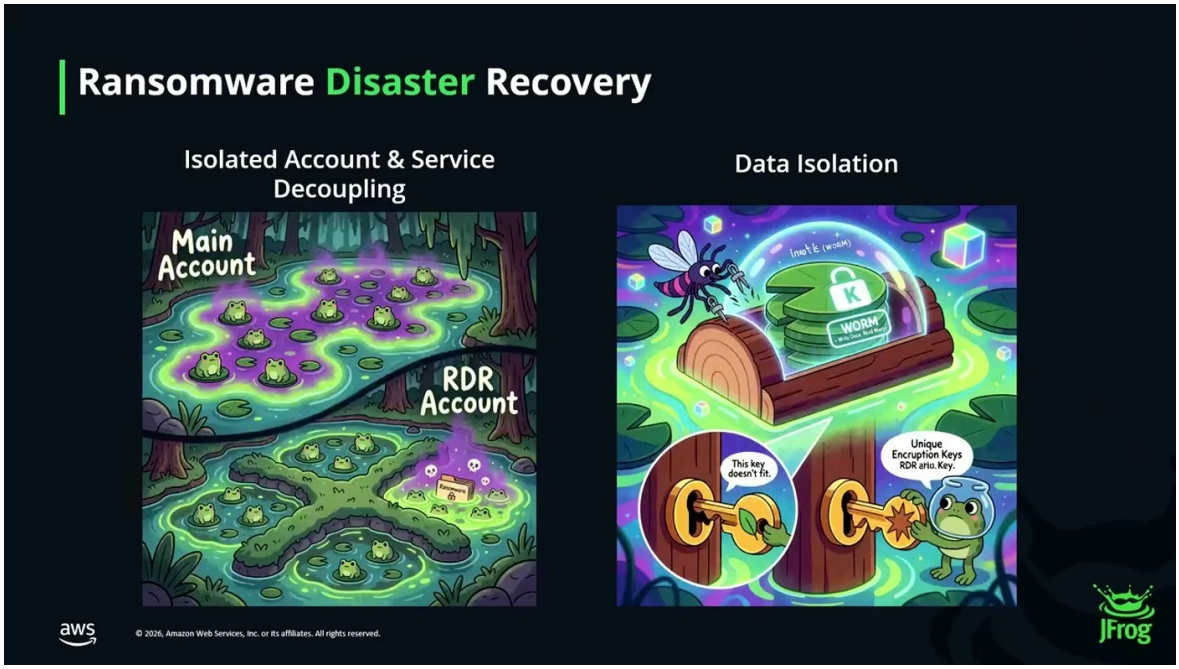
אפשרות	מה היא עולה
שחזור snapshot	אובדן נתונים גדול — ה-snapshot נלקח פעם ביום; ואחריו צריך לבנות מחדש את כל ה-global cluster, להוסיף את ששת הריג'נים ולסדר את ה-endpoints
Point-in-Time Recovery	פחות אובדן נתונים, כי בוחרים את נקודת הזמן — אך אותה פרוצדורת בנייה מחדש

אפשרות	מה היא עולה
תיקון כירורגי ישיר	מהיר מאוד, אך חושף ל-user error: טעות ידנית מופצת לכולם באותה מהירות
תיקון דרך האפליקציה	האיטי ביותר, והמומלץ — קוד שנבדק ושבטוחים שיעבוד; "הסיכון יהיה כמה שיותר נמוך"

המסקנה שנאמרה מפורשות: "כנראה שהפתרון הכי טוב יהיה תיקון דרך האפליקציה" [30:39]. Aurora Fast Clone אינו פותר את הבעיה — הדאטה בקלון עדיין יהיה משובש, שכן הוא מייצר compute עם מצביע לאחסון המקורי — אך הוא כן משמש לבדיקת התיקון עצמו לפני שמריצים אותו בפרודקשן [30:39].

13. אתגר ב': Ransomware Disaster Recovery

ההבדל בין DR רגיל ל-ransomware DR הוגדר בחדות: "אנחנו לא מאבדים פה ריג'ן אחד" [32:10] — כל החשבון תחת מתקפה, וכל מי שנמצא בתוכו נמצא בסיכון לשחיתות נתונים.



שני העקרונות: הפרדת חשבון ופירוק תלות בין שירותים, לצד בידוד הדאטה

הרעיון: להעתיק את הדאטה לחשבון RDR מבודד שלאיש אין גישה אליו — חוץ ממערכות הגיבוי האוטומטיות — ולהצפין אותו במפתח שונה מזה של ה-main account, שכן ה-main account הוא זה שנחשב compromised. באותו חשבון גם מפצלים את ה-blast radius בחלוקת המשאבים [32:10].

שלושת המסלולים: CMK ישיר, ומעליו שתי חלופות לדאטה שהוצפן במפתח ברירת המחדל

המסלול התקין — Customer Managed Key

אם הדאטה ב-Aurora מוצפן ב-CMK, "שזו הדרך התקינה", המצב פשוט: "אנחנו נייצר וולט ב-RDR Account ובעצם אנחנו נפעיל את AWS Backup Service שיפעיל גיבוי קרוס אקאונט של הדאטה מהמיין ל-[32:10] "RDR, לתוך vault שמוגדר כ-immutable.

כשהדאטה הוצפן במפתח ברירת המחדל

"אבל יש מקרים פחות נעימים שבהם הצפנתם את הדאטה עם KMS, שהוא ה-key הדפולטי של AWS, זה קורה מדי פעם, legacy, לא שמו לב" [32:10]. שתי חלופות הוצגו.

הראשונה דורשת downtime: להעביר את הקלאסטר הראשי לעבוד עם CMK — ליצור קלאסטר חדש מוצפן ב-CMK, לקחת snapshot מהראשי, לשחזר אליו, ואז לעבור למסלול הראשון [33:48].

השנייה, ללא downtime: "אנחנו יכולים גם לייצר לוקל וולט נוסף במיין אקאונט שהוא מוצפן עם קסטמר מנגי" [33:48]. AWS Backup מעתיק תחילה את הדאטה ל-vault המקומי ומצפין אותו מחדש ב-CMK, ורק אחר כך מבצע cross-account copy ל-vault שב-RDR, שם הוא מוצפן שוב ב-CMK של חשבון ה-RDR.

האם כדאי Logically Air-Gapped Vault

ההצגה של LAG Vault הייתה מאוזנת ולא שיווקית. בצד האחד: האחסון נמצא בחשבון של AWS ולא בשלכם, הדאטה immutable by default, ואפילו ה-root account שלכם אינו יכול לגעת בו [35:23]. בצד האחר: המפתח שמצפין אותו נעול על ידי AWS, כך שהשליטה על ההצפנה אינה עוד בידיכם; כדי לאפשר restore לחשבון ה-RDR בעת אירוע צריך להעניק לחשבון של AWS הרשאות מתאימות; ו-"ה-retention time הוא שבע ימים, ואי אפשר לעשות retention time שהוא פחות מזה" [35:23].

> **DevOps Director, JFrog, [35:23] —** אבל את כל הקונפיגורציות האלה אפשר גם לקנפג בעצמנו בתוך הוולט שקיים באקאונט שלנו ואז יש לנו קצת יותר שליטה על הרטנשן טיימפ

המסקנה שלה: אם יש דרישת compliance מפורשת ל-vault בלתי-משתנה ברמת ציות גבוהה — ייתכן ש-LAG Vault מתאים. אחרת, vault בחשבון שלכם עם אותן הגדרות נותן יותר שליטה.

14. מה לוקחים מכאן

- **לתרגם את יעדי ה-RTO/RPO לתרחישים ולא לכותרת אחת.** הסשן מספק ארבעה מספרים נפרדים: נפילת AZ, נפילת writer, מעבר ריג'ן מתואם, ומעבר ריג'ן כפוי. יעד גורף של "פחות מדקה" מסתיר את העובדה שרק failover כפוי כרוך באובדן נתונים.

- **להגדיר failover priority מראש, לא בזמן אירוע.** כששוקלים replicas בגדלים שונים, ה-tier הוא מה שמבטיח שה-replica שתקודם תעמוד בעומס [9:15]. JFrog מציגה זאת כהגדרה קיימת ולא כהחלטה בזמן אמת [27:36].
- **לבדוק אם המסלול היקר של השדרוגים עדיין מופעל ידנית.** שילוב Blue/Green עם Fast Clone נותן סביבת בדיקה זולה ומעבר של שניות; Upgrade Rollout Policy הופך שדרוג של מאות דאטאבייסים לתהליך מתועד בגלים.
- **לוודא באיזה מפתח מוצפנים הדאטאבייסים היום.** זו הנקודה המעשית ביותר בפרק ה-ransomware: מי שהצפין במפתח ברירת המחדל של AWS נדרש למסלול עוקף — או downtime, או שרשרת vaults נוספת.
- **לזכור שהתבניות אינן מכסות שחיתות נתונים לוגית.** רפליקציה מהירה מפיצה טעות באותה מהירות שבה היא מפיצה תיקון. התשובה שניתנה מהבמה אינה טכנולוגית אלא תהליכית: תיקון דרך האפליקציה, אחרי בדיקה על clone.
- **לשקול headless clusters בריג'ני DR.** אחסון בלבד, בלי compute, עם רפליקציה ברמת האחסון — חיסכון ישיר בעלות עבור ריג'נים שאינם משרתים תעבורה בשגרה [13:51].

מונחון

מונח	מה זה
RTO	Recovery Time Objective — פרק הזמן שלוקח להתאושש מתקלה
RPO	Recovery Point Objective — פרק הזמן המקסימלי שבו מותר לאבד מידע
Aurora Global Database	יכולת לייצר עד 15 קלאסטרים משניים בריג'נים שונים, ברפליקציה אסינכרונית
Headless cluster	קלאסטר משני עם אחסון בלבד, בלי compute — רפליקציה ברמת האחסון בלי עלות מחשוב
Global endpoint	נקודת חיבור שמצביעה תמיד אל ה-primary cluster הנוכחי, בכל ריג'ן שהוא
Failover	מעבר חד-צדדי ויזום של ה-primary לריג'ן אחר; עלול לאבד נתונים בגובה ה-replication lag
Switchover	מעבר מתואם בין שני הקלאסטרים באמצעות marker ב-transaction log; RPO אפס
Failover tier	עדיפות מספרית לכל replica — הנמוך יותר מקודם ראשון
Aurora Capacity Unit	יחידת קיבולת ב-Aurora Serverless, שוות ערך לשני גיגהבייט
Fast Clone	שכפול מהיר של דאטאבייס — compute חדש עם מצביע לאחסון המקורי
Blue/Green Deployment	קלאסטר ירוק מקביל, מסונכרן ב-logical replication, שאליו מעבירים תעבורה אחרי בדיקות
Upgrade Rollout Policy	מדיניות ברמת AWS Organizations לשדרוג דאטאבייסים בגלים לפי תגיות
RDS Proxy	connection pool מנוהל; מקצר זמן מעבר ומחליף disconnect בעלייה ב-latency
AWS Advanced Wrapper Drivers	ספריית קוד פתוח שעוטפת דרייברים ומקצרת changeover בלמעלה מ-60%
CMK	Customer Managed Key — מפתח הצפנה בשליטת הלקוח, בניגוד למפתח ברירת המחדל של AWS
LAG Vault	Logically Air-Gapped Vault — מאגר גיבוי immutable המאוחסן בחשבון של AWS, retention מינימלי של 7 ימים
RDR Account	חשבון מבודד לשחזור מאירוע כופר, ללא גישה אנושית ועם מפתח הצפנה נפרד