digital whisper
Technologic Papers

# שימוש במתודולוגיית DevOps-להטמעת עדכוני תוכנה בארגון

מאת יובל סיני

## מבוא

מערכות המחשוב מהוות כלי עזר רב תכליתי לארגונים, והלכה למעשה נדיר לראות ארגון אשר מצליח כיום למלא את תכליתו כראוי ללא שימוש במערכות המחשוב. עם זאת, השימוש במערכות המחשוב מחייב תחזוקה שוטפת, דבר הכולל בין השאר שדרוג גרסאות תוכנה, התקנת עדכוני תוכנה לתיקוני באגים ובעיות אבטחה (פאצ'ים), וכדומה.

כמו כן, עקב החשיפה הגבוהה לאיומים בתחום אבטחת המידע הגנת הפרטיות, ניתן לזהות כי הצורך בהתקנת עדכוני תוכנה לתיקון פגיעויות ידועות גובר והולך, דבר המחייב את הארגונים להשקיע תשומות רבות לשם עמידה בקצב העדכון המומלץ.

בנוסף, ארגונים אשר כפופים לתקני אבטחת מידע, כדוגמת תקן PCI ו-ISO נדרשים לנהל תהליך הערכת סיכונים תקופתי אשר כולל קביעת מדיניות ישימה להטמעת עדכוני תוכנה לתיקון בעיות אבטחה (פאצ'ים).

ולהלן מצ"ב דוגמא להתייחסות תקן PCI[1] לנושא הטמעת עדכוני תוכנה לתיקון בעיות אבטחה (פאצ'ים):

> "Ensure that all system components and software are protected from known vulnerabilities by installing applicable vendor-supplied security patches. Install critical security patches within one month of release."

דוגמא נוספת הינה התייחסות תקן ISO 27002 (2013) לנושא הטמעת עדכוני תוכנה לתיקון בעיות אבטחה (פאצ'ים):

**12.6.1 Management of technical vulnerabilities**

Control

Information about technical vulnerabilities of information systems being used should be obtained in a timely fashion, the organization's exposure to such vulnerabilities evaluated and appropriate measures taken to address the associated risk.

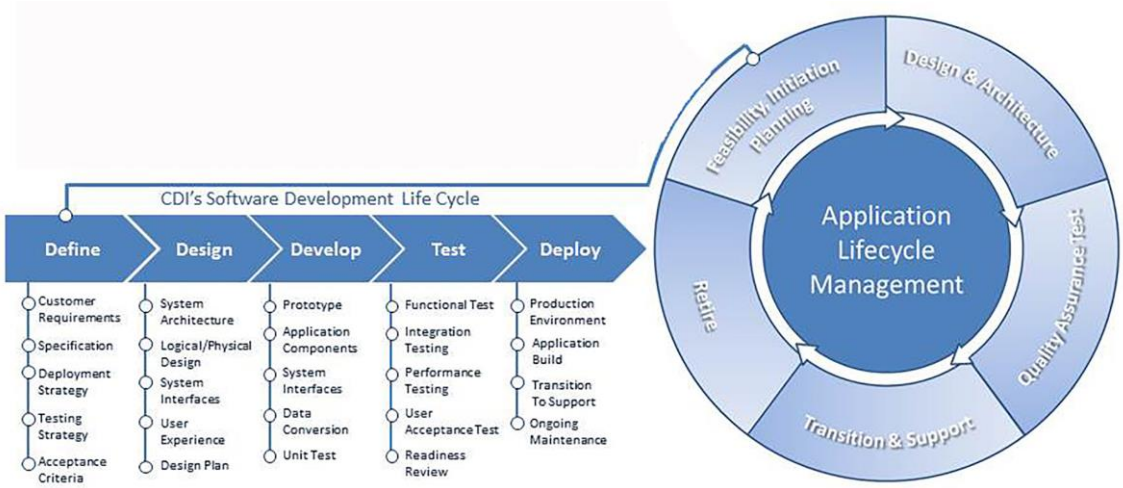
[1] סעיף 6.2 תקן PCI גרסה 3.0

אקדים את המאוחר ואציין כי נושא שדרוג גרסאות תוכנה, התקנת עדכוני תוכנה לתיקוני באגים ובעיות אבטחה (פאצ'ים) וכו' נכלל בד"כ תחת שלב "התחזוקה" (Support)[2] במחזור חיי תוכנה ( Application Lifecycle Management, ובראשי תיבות ALM).

## "מחזור חיי תוכנה" (Application Lifecycle Management)

לשם הרחבת היריעה בנושא "מחזור חיי תוכנה" אשתמש בדוגמא מייצגת - "מחזור חיי תוכנה" בחברת CDI Corporation. עם זאת, ברצוני לציין כי המידע המוצג בנושא "מחזור חיי תוכנה" הינו בסיסי, וכי מדובר בנושא מורכב הכולל בחובו תפיסות-אסטרטגיות-פילוסופיות מגוונות.

ראשית כל, ניתן ללמוד מהמושג "מחזור חיי תוכנה" ומהתרשים המצ"ב כי התהליך הינו מחזורי, וכי הוא כולל בחובו מספר שלבים. כמו כן, דבר נוסף שבולט הינו שהצלחת התהליך אינה תלויה בגורמי IT (Information Technology) בלבד, אלא מדובר בתהליך רוחבי. כך לדוגמא, שלב איסוף הדרישות מחייב הכרה מעמיקה עם קהל הלקוחות ואיתור צרכי הלקוחות, תוך התאמת האסטרטגיה הארגונית לשם מתן מענה אופטימלי לצרכי הלקוחות.




עם זאת, התפיסות המקוריות בנושא "מחזור חיי תוכנה" התמקדו בלקוחות חוץ אירגוניים בלבד, ולפיכך שכיח לראות במרבית הארגונים נתק מסוים בין צוות הפיתוח (Developers Team) לצוות מערכות המידע (Information Technology), דבר אשר בד"כ פוגע באפקטיביות הארגונית.

[2] המושג "מחזור חיי תוכנה" מכיל בחובו מספר רב מתודולוגיות וגישות, וראוי לציין כי חלק מהתודולוגיות והגישות משתמשות בטרמנולגיה שונה במקצת.

## מבוא ל-DevOps

לשם הצגת מתודולוגיית DevOps אציג את הגדרתה[3] של Margaret Rouse למושג DevOps, כפי שהוא מופיע באתר WhatIs.com:

"DevOps מהווה מיזוג בין המשימות המבוצעות על ידי צוותי פיתוח (Developer Team) האחראים לפיתוח היישומים בחברה, לבין צוותי התפעול (Operation Team) של המערכות השונות בארגון. המושג DevOps ניתן להצגה ע"י שימוש במספר פרשנויות. הפרשנות הרחבה יותר גורסת כי ה-DevOps מהווה גישה פילוסופית או גישה תרבותית אשר מטרתה לקדם תקשורת טובה יותר בין שתי קבוצות או יותר, כך שאלמנטים (ומשימות) משותפים בין הקבוצות יעשו באופן אוטומטי, וזאת בהתאם לתכנון מוסכם מראש.

הפרשנות הצרה יותר גורסת כי DevOps הוא תיאור תפקיד עבור עובד\ת המחזיק ביכולות וכישורים לעבוד הן כמפתח (Developer) והן כמהנדס מערכת (Systems Engineer). בתעשיות מסוימות המושג משמש לתאר מתווך בין שתי קבוצות (או יותר) המשמש כ-Scrum Master[4] אשר מסייע לצוותי פיתוח וצוותי התפעול לממש "מחזור חיי תוכנה" בארגון באופן הולם.

הצורך בהתרת חסמים בין צוות הפיתוח לצוות התפעול זוכה לחיזוק עקב הופעת ה"מחשוב ענן" (Cloud Computing) והופעת ה- SDN (Software-Defined Networking).

ראוי לציין כי מתודולוגית ה-DevOps שונה מהמתודולוגיה המסורתית בארגונים אשר גרסה כי צוות הפיתוח אחראי לאיסוף דרישות עסקיות לשם פיתוח התוכנה, ולאחר מכן לכתיבת הקוד. כמו כן, המתודולוגיה המסורתית גרסה כי צוות הפיתוח (Developers Team) אחראי לבדיקת תוצרי התוכנה בסביבה מבודדת אשר מטרתה לשמש כסביבת QA (Quality Assurance) - ולאחר שהדרישות העסקיות והטכנולוגיות נענו, מתבצע שחרור קוד התוכנה לצוות התפעול (Operation Staff), או לחילופין במקרה של העדר עמידה בדרישות העסקיות ו\או הדרישות הטכנולוגיות ישנו צורך לחזור "לשולחן השרטוטים". רק לאחר הצלחת שלב זה, צוות התפעול (Operation Staff) פורס את רכיבי התוכנה והוא זה שאחראי על תחזוקתה."

לפיכך, ניתן לראות כי ארגונים אשר אימצו כראוי את מתודולוגיית ה-DevOps מצליחים כיום לשפר את פעילותם ביחס למתחרים, ובכך יש ביכולתם למנף את היתרונות הטכנולוגיים ליתרונות עסקיים. כמו כן,

[3] התרגום אינו נאמן למקור אחד לאחד, וכולל בחובו שינויים קלים אשר מטרתם להקל על הקריאה.

[4] Scrum Master מהווה גורם בארגון שהינו בעל ראיה הוליסטית, ויכולת גישור והובלה של הצוותים השונים לשם השגת המטרות הארגוניות הרצויות (כדוגמת לו"ז, איכות תוצר נדרש). עם זאת, ה Scrum Master אינו מהווה מנהל פרויקט, אלא מהווה תפקיד יעודי. תפקיד זה שכיח בארגונים אשר מאמצים את גישת Agile בעת ניהול פרויקטי תוכנה.

שכיח לראות כי ארגונים אשר אימצו את מתודולוגיית ה-DevOps, מאמצים אף את גישת ה- Continuous Deployment. רוצה לומר, מאמצי מתודולוגיית ה-DevOps וגישת ה-Continuous Deployment דוגלים באימוץ מדיניות ארגונית אשר כוללת ביצוע עדכונים בתדירות גבוהה (לעיתים אף מספר פעמים ביום) לרכיבי תוכנה בסביבת יצור, דבר אשר כולל בנוסף תהליכי למידה והשתפרות מתמדת (Continuous Improvement).

## Security within Continuous Deployment

כפי שצוין קודם לעיל, גישת ה-Continuous Deployment דוגלת בביצוע עדכונים בתדירות גבוהה (לעיתים אף מספר פעמים ביום) לרכיבי תוכנה בסביבת יצור. לפיכך, ניתן לראות כי הגישה כוללת אף התקנת עדכוני תוכנה לתיקוני באגים ובעיות אבטחה (פאצ'ים) בתדירות גבוהה בסביבת היצור.

## שימוש במתודולוגיית-DevOps להטמעת עדכוני תוכנה בארגון

התקנת עדכוני תוכנה בתדירות גבוהה בסביבת היצור מחייבת אימוץ מתודולוגיה מתקדמת (כדוגמת ה-DevOps) אשר יש באפשרותה לסייע לארגונים להתגבר על אתגרים שכיחים, כדוגמת:

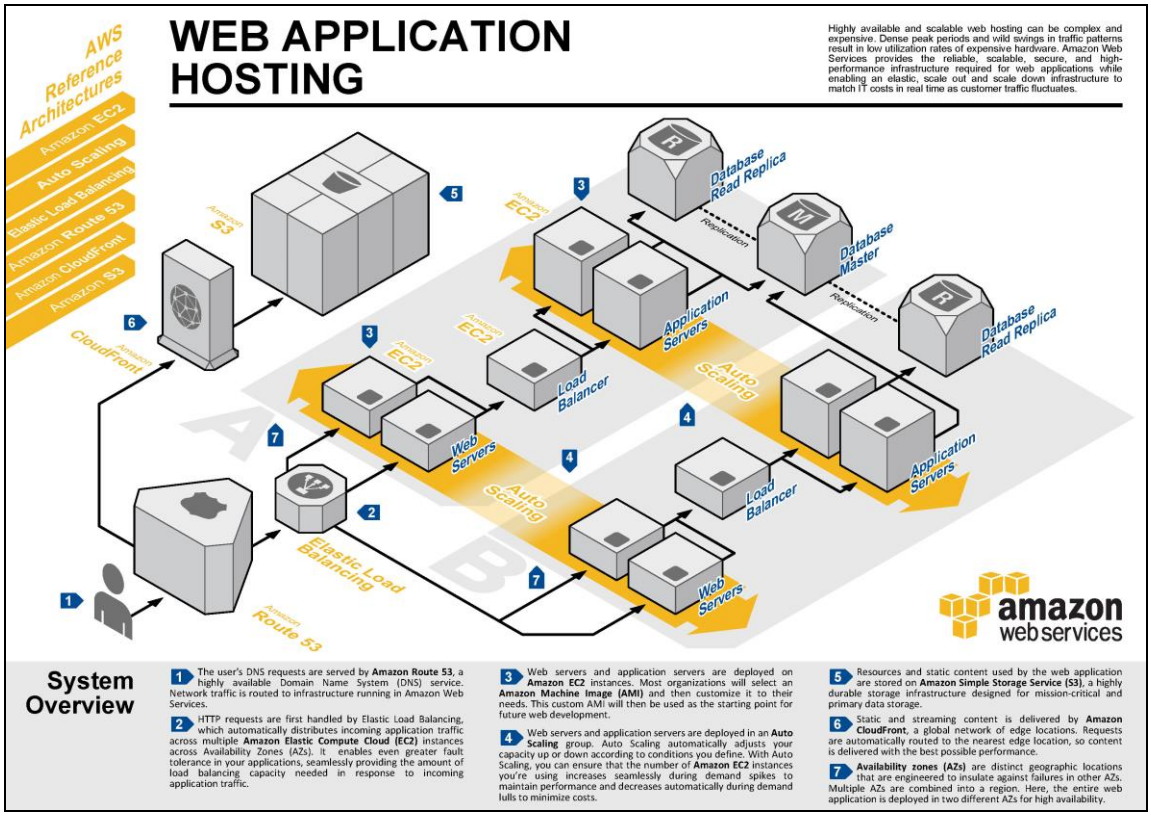
א. **חלון השבתה** (Maintenance Window) - חלון ההשבתה מהווה זמן שבו ניתן לבצע השבתה של מערכות יצוריות. עם זאת, ניתן לראות כי בארגונים רבים חלון ההשבתה עומד על דקות ספורות בחודש, אם כלל קיים.

ב. **צמצום הסיכונים** (Reducing Risks) - ביצוע עדכון לרכיבי תוכנה כולל בחובו סיכונים תפעוליים ועסקיים, כדוגמת היתכנות לקיומן של בעיות תאימות, באגים, וכדומה. עם זאת, הארגון נדרש למצוא את הדרך האופטימלית אשר תמנע חשיפה גבוהה לסיכונים הנובעים מביצוע העדכונים התכופים.

ג. **תשומות** - ביצוע עדכונים לרכיבי תוכנה מחייב השקעה ניכרת בתשומות, כדוגמת כ"א, סביבת בדיקות, וכדומה. יש לזכור כי בניגוד לעבר, תקציבי ה-IT קטנים עם הזמן, אך מאידך גיסא, הדרישות הארגוניות-עסקיות גדלות. ובמילים אחרות, ארגונים נאלצים כיום לפעול בהתאם לגישת Do More With Less.

לשם המחשת השימוש במתודולוגיית-DevOps להטמעת עדכוני תוכנה בארגון אשתמש בפתרון לדוגמא הניתן למימוש ב-Amazon Web Services[5] בעמוד הבא.

[5] הבחירה בשירות Amazon Web Services נעשתה לשם הנוחות, ואין בבחירה זו להעיד על טיב השירות \או הפתרון, ובכלל זה להתאמתו של השירות לארגון כזה או אחר.




כבר בעיון ראשוני בתרשים הנ"ל ניתן ללמוד מספר דברים חשובים על ארכיטקטורת הפתרון:

א. הפתרון בנוי משכבות מודולריות והיררכיות.

ב. בכל שכבה מודולרית והיררכית הרכיבים בשכבה מוטמעים בתצורה שרידה - Active\Active.

ג. הפתרון מכיל בקרת גישה ברמת DNS ו-IP, ובכלל זה מימוש Load Balance ו-Caching אלסטיים.

ד. הפתרון מכיל מערכת ניטור ובקרה מרכזית.

כמו כן, בעיון בתיעוד נלווה ניתן ללמוד כי הפתרון כולל שימוש ב-Immutable Servers[6], כאשר המושג Immutable Servers נגזר במקור ממתודולוגית ה-DevOps. כזכור, מתודולוגית ה-DevOps רואה את הצורך לביצוע אינטגרציה בין יכולות וצרכים של קבוצות שונות בארגון, וזאת לשם שיפור יכולת הארגון לעמוד ביעדים אשר הנהלת הארגון קבעה.

לפיכך, ה-Immutable Servers מהווים מעין "יחידות עבודה" אוטונומיות הניתנות להחלפה בצורה דינמית. כך לדוגמא, במקום להשתמש בגישה המסורתית בה צוות התפעול בארגון מתקין עדכוני אבטחה (פאצ'ים) על מערכת הפעלה של השרת (דבר המחייב השבתה תפעולית), השימוש ב- Immutable Servers מאפשר לבצע "החלפה חמה" בין שרת בעל מערכת הפעלה לא מעודכנת, ל-Image שרת המכיל מערכת הפעלה מעודכנת (בסביבות וירטואליות זמן ההטמעה עומד על דקות בודדות).

[6] את המושג Immutable Servers הטביע Ben Butler-Cole.

לפיכך, ניתן לאתר את השלבים העיקריים בעת ביצוע "החלפה חמה" (Hot Replacing) בין שרת בעל מערכת הפעלה לא מעודכנת, ל-Image שרת המכיל מערכת הפעלה מעודכנת:

א. צוות הפיתוח משחרר גרסת עדכון לרכיב כזה או אחר.

ב. צוות התפעול מכניס את השרת הקיים ב-Servers Pool ל"שלב ייבוש" (Drying Phase). בשלב זה לא השרת לא נדרש לספק שירות ללקוחות חדשים, ולאחר סיום פעילות של הלקוחות הקיימים השרת זמין לתחזוקה.

ג. צוות התפעול מסיר את השרת מה-Servers Pool, ובמקום השרת הישן צוות התפעול מטמיע Image שרת עדכני.

ד. צוות התפעול מחזיר את השרת (המעודכן) ל-Servers Pool. עם זאת, השרת עדיין לא מוגדר למתן שירות ללקוחות.

ה. צוות התפעול מתיר גישה לשרת (המעודכן) באופן מדורג: צוות QA, כלי בדיקה אוטומטים, לקוחות פיילוט.

ו. לאחר הצלחת שלב ה', צוות התפעול מתיר גישה לשרת (המעודכן) באופן מדורג ללקוחות הארגון. הגישה המדורגת מאפשרת איתור מבעוד מועד של כשלים אפשריים, ובכלל זה איתור בעיות Performance שלא אותרו בשלב ה'. בסופו של התהליך השרת (המעודכן) מוחזר לשירות באופן מלא.

אחד היתרונות הבולטים בשימוש ב-Immutable Servers הוא העדר הצורך בסביבות נפרדות לטובת QA וייצור. כמו כן, ניתן לראות כי זמן ההטמעה בפועל הינו קצר משמעותית ביחס למתודולוגיות מסורתיות, וכי ניתן להטמיע עדכונים בתדירות גבוהה. עם זאת, הצלחת הפתרון מחייבת אימוץ גישה רוחבית אחידה, הכוללת שיתוף פעולה הדוק בין הגורמים השונים בארגון. כמו כן, לשם הצלחת הפתרון ישנו צורך לעבוד עם כלי ניהול תצורה אוטומטיים (Automated Configuration Tools), כדוגמת CFEngine[7], המאפשרים אוטומציה מלאה של התהליך. כך לדוגמא, תהליך הוספת חוקי Firewall וביצוע הגדרות Load Balance אמורים להתבצע באופן אוטומטי, וללא צורך בהתערבות ידנית.

## סיכום

"מחזור חיי תוכנה" (Application Lifecycle Management) מהווה תהליך חשוב בארגונים המסתמכים על מערכות המחשוב. השימוש במתודולוגית DevOps וגישת Continuous Deployment יכולים לשפר את יכולתו של הארגון להטמיע עדכוני תוכנה בתדירות גבוהה, תוך צמצום הסיכונים הנובעים מהטמעת

[7] הבחירה בכלי האוטומציה וניהול התצורה CFEngine נעשתה לשם הנוחות, ואין בבחירה זו להעיד על טיב הכלי ו\או הפתרון, ובכלל זה להתאמתו של הכלי לארגון כזה או אחר.

עדכונים אלו. כמו כן, ניתן למנף את יכולת זו לשם עמידה בדרישות העסקיות של הארגון, ובכלל זה עמידה בדרישות ל-Security Compliance.

"If you want to build a ship, don't drum up the men to gather wood, divide the work, and give orders. Instead, teach them to yearn for the vast and endless sea ".

Antoine de Saint-Exupéry

## על המחבר

יובל סיני הינו מומחה אבטחת מידע, סייבר, מובייל ואינטרנט, חבר קבוצת SWGDE של משרד המשפטים האמריקאי.

# ביבליוגרפיה

## ביבליוגרפיה כללית:


- Tracking the Progress of an SDL Program Lessons from the Gym, Cassio Goldschmidt, OWASP, 2010
- Microsoft Security Development Lifecycle for IT, Rob Labbé
- Architecting for The Cloud: Best Practices - Jinesh Varia, 2011 http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf


## ביבליוגרפיה בנושא מחזור חיי תוכנה \ Application Lifecycle Management (ALM):


- מודלים של מחזור חיי תוכנה Software Life-Cycle Models: http://webcourse.cs.technion.ac.il/236321/Winter2011-2012/ho/WCFiles/unit03-process-models-1112.pdf
- Scrum: a Breathtakingly Brief and Agile Introduction, Chris Sims, Dymaxicon, 2012
- CDI's Application Lifecycle Management (ALM): http://www.cdi-ps.com/technology/application-lifecycle-management/


## ביבליוגרפיה בנושא DevOps:


- Rethinking building on the cloud: Part 4: Immutable Servers http://www.thoughtworks.com/insights/blog/rethinking-building-cloud-part-4-immutable-servers
- The Codeship Workflow: Developing a new feature http://blog.codeship.io/2013/08/16/the-codeship-workflow-part-1-developing-a-new-feature.html
- ImmutableServer http://martinfowler.com/bliki/ImmutableServer.html
- Continuous Delivery and DevOps FAQs, Paul Swartout, January 2013 http://www.packtpub.com/article/continuous-delivery-devops-faqs
- People, Process & Tools - The Essence of DevOps, Richard Campbell
- DevOps and Security: It's Happening. Right Now, Helen Bravo

- The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win, Gene Kim, George Spafford, Kevin Behr, IT Revolution Press, 2013
- DevOps
  http://searchcloudcomputing.techtarget.com/definition/DevOp


### ביבליוגרפיה בנושא תקינה ורוגלציה:


- Payment Card Industry (PCI) Data Security Standard
  Requirements and Security Assessment Procedures, Version 3.0, November 2013
  https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf
- The Journey to ISO 27001 (Part 1), Ricky M. Magalhaes, Published on 3 July 2013
  http://www.windowsecurity.com/articles-tutorials/misc_network_security/journey-iso-27001-part1.html
- The Journey to ISO 27001 (Part 2), Ricky M. Magalhaes, Published on 28 Aug. 2013
  http://www.windowsecurity.com/articles-tutorials/misc_network_security/journey-iso-27001-part2.html
- ISO 27002 (2013 Second Edition)
  http://www.scribd.com/doc/177330349/ISO-IEC-27002-2013