DEVOPS TRAINING COURSE

מרצה: אלכס גורבצ'וב

לו"ז שלנו

משך הקורס:

3.02.2025

24.02.2025

ימי לימוד:

יום ב'

9:00-13:00

היקף שעות:

16 שעות

מספר מפגשים:

4 מפגשים

סדר יום

חלק 1 - 9:00-10:15

הפסקה - 10:15-10:25

חלק 2 - 10:25-11:35

הפסקה - 11:35-11:45

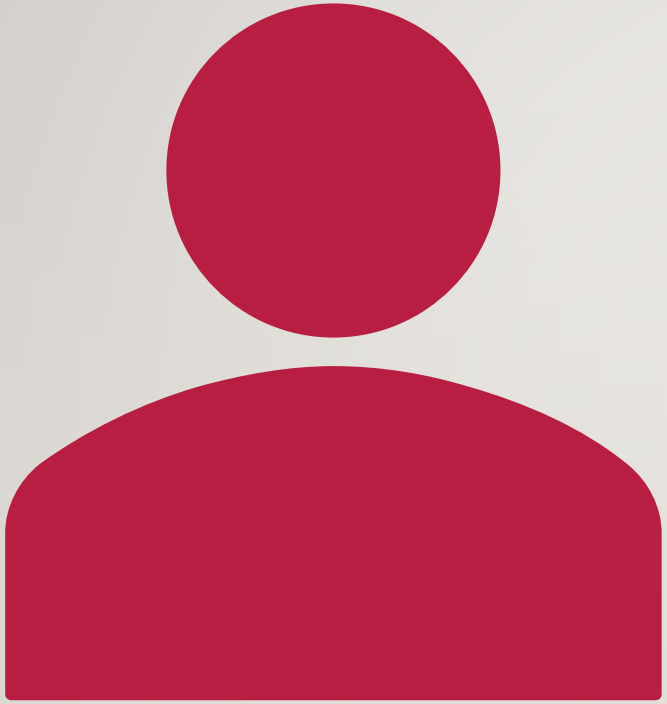
חלק 3 - 11:45-13:00

נושאי השיעור

- Introduction to DevOps
  - DevOps principles and practices
  - Development lifecycle
  - Tools overview
- Cloud Native Fundamentals
  - Cloud Native Applications architecture
  - The CNCF Landscape
  - Microservices principles
  - 12-Factor App methodology

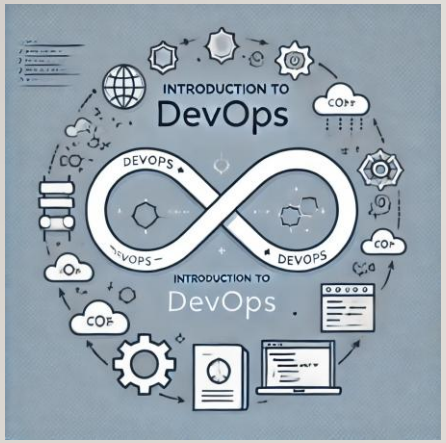
נושאי השיעור

- Docker Fundamentals
  - Container basics
  - Docker architecture
  - Docker CLI
- Dockerfile & Docker Compose
  - Creating custom images
  - Multi-container applications
  - Best practices



INTRODUCTION TO DEVOPS

INTRODUCTION TO DEVOPS



מהו DevOps?

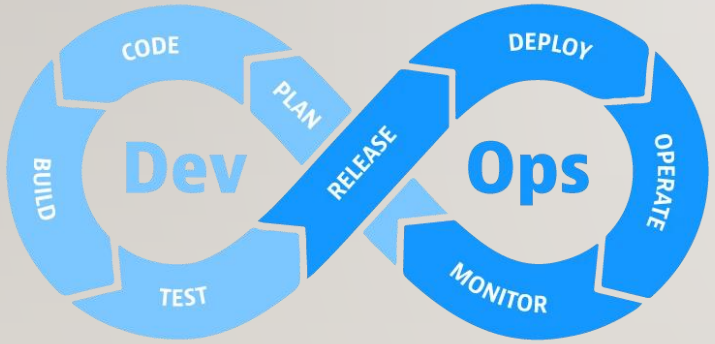
DevOps = שילוב של פיתוח (Development) ותפעול (Operations).

הרעיון המרכזי הוא לקרב בין מפתחים לבין אנשי תפעול כדי להאיץ תהליכים, לשפר את איכות התוכנה ולשחרר עדכונים בצורה מהירה ובטוחה.

דוגמה:

- **תרחיש**: מהנדס מפתח רכיב חדש לשרת. הוא מסיים את הפיתוח ושולח אותו לבדיקות. צוות הבדיקות מוצא באג קריטי שמונע מהרכיב לעבוד כראוי.
- **בעיה**: הבאג היה יכול להתגלות מוקדם יותר, במהלך הפיתוח, אם המהנדס היה משתמש בכלים ובשיטות של DevOps.
- **פתרון**: DevOps מעודד אוטומציה של בדיקות ואינטגרציה רציפה, מה שמאפשר לגלות בעיות מוקדם יותר ולתקן אותן בזמן.

INTRODUCTION TO DEVOPS



יתרונות

- **שיפור שיתוף הפעולה בין צוותים**: DevOps מעודד שיתוף פעולה ותקשורת בין צוותי הפיתוח והתפעול, מה שמוביל להבנה טובה יותר של הצרכים והאתגרים של כל צוות וליצירת פתרונות יעילים יותר.
- **הפחתת עלויות:** DevOps יכול לסייע בהפחתת עלויות על ידי אוטומציה של תהליכים, שיפור יעילות השימוש במשאבים ומניעת תקלות.
- **שיפור איכות המוצר:** DevOps מאפשר זיהוי ותיקון בעיות מוקדם יותר בתהליך הפיתוח, מה שמוביל למוצרים איכותיים יותר.
- **שיפור חוויית הלקוח:** DevOps מאפשר פריסה מהירה יותר של עדכונים ותכונות חדשות, מה שמשפר את חוויית הלקוח.
- **הגברת החדשנות:** DevOps מאפשר לצוותים להתנסות בטכנולוגיות חדשות ולפתח מוצרים חדשניים יותר.

בנוסף DevOps, מציע יתרונות רבים נוספים שיכולים לסייע לארגונים לשפר את ביצועיהם ולשגשג בסביבה העסקית התחרותית של היום.

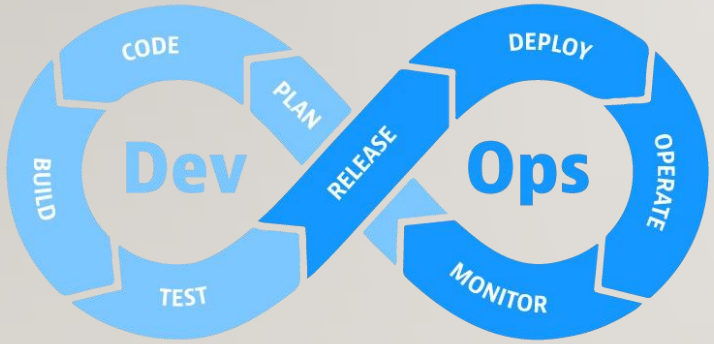
INTRODUCTION TO DEVOPS

למה DevOps חשוב?

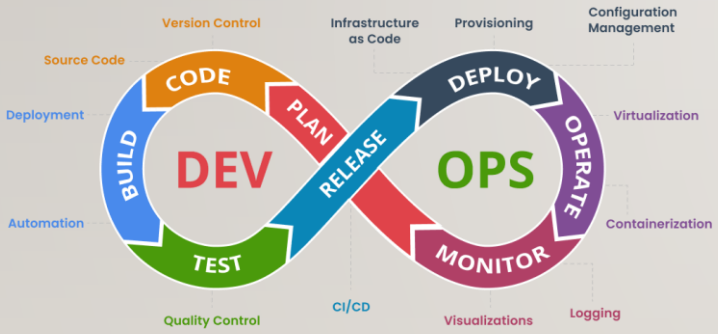
בעולם של היום, חברות כמו Netflix, Amazon ו-Google מעדכנות את המערכות שלהן מאות פעמים ביום! בלי DevOps, זה היה בלתי אפשרי.

דוגמה:

Netflix מריצה אלפי שינויים ביום - אם כל עדכון היה מצריך עצירה של השירות, אף אחד לא היה משתמש בו.



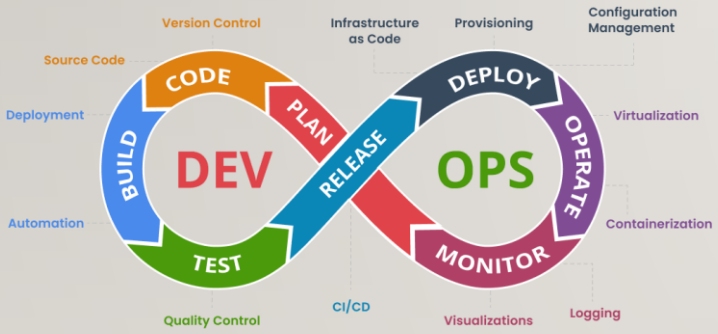
INTRODUCTION TO DEVOPS



עקרונות DevOps

- **אוטומציה (Automation)**. אוטומציה היא מרכיב מפתח ב-DevOps. היא נועדה להחליף תהליכים ידניים וחוזרים בתהליכים אוטומטיים, מה שמאפשר:
  - **הפחתת זמן הפיתוח**: אוטומציה של בדיקות, פריסות ופעולות נוספות חוסכת זמן יקר ומאפשרת לצוותים להתמקד במשימות מורכבות יותר.
  - **שיפור איכות התוכנה**: אוטומציה של בדיקות ואינטגרציה רציפה (CI) מסייעת לזהות ולתקן באגים מוקדם יותר בתהליך הפיתוח.
  - **הפחתת עלויות**: אוטומציה של תהליכים מפחיתה את הצורך במשאבים אנושיים ומפחיתה את הסיכון לטעויות אנוש.

INTRODUCTION TO DEVOPS



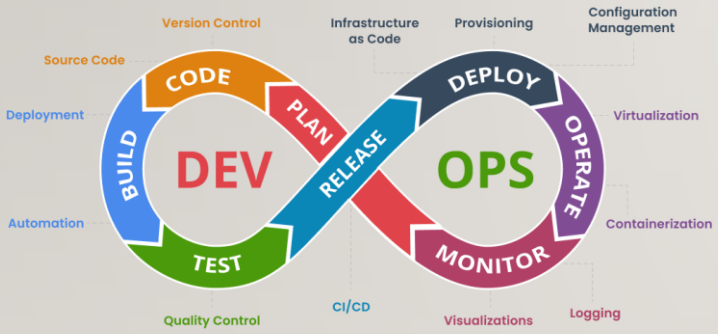
עקרונות DevOps

- **שילוב רציף ופריסה רציפה (CI/CD)**. CI/CD הם שני עקרונות מרכזיים ב-DevOps:
  - **שילוב רציף (Continuous Integration):** תהליך אוטומטי של אינטגרציה של שינויים בקוד ממפתחים שונים, בדיקות אוטומטיות ובניית גרסאות של התוכנה.
  - **פריסה רציפה (Continuous Delivery/Deployment):** תהליך אוטומטי של פריסת גרסאות חדשות של התוכנה לסביבות שונות (בדיקות, ייצור).

CI/CD מאפשרים:

- **קצב פיתוח מהיר יותר:** אינטגרציה ובדיקות אוטומטיות מאפשרות לצוותים לשחרר גרסאות חדשות בתדירות גבוהה יותר.
- **איכות תוכנה גבוהה יותר:** בדיקות אוטומטיות ופריסות מבוקרות מפחיתות את הסיכון לבעיות בייצור.
- **זמן תגובה מהיר יותר לבעיות:** CI/CD מאפשרים לזהות ולתקן בעיות במהירות רבה יותר.

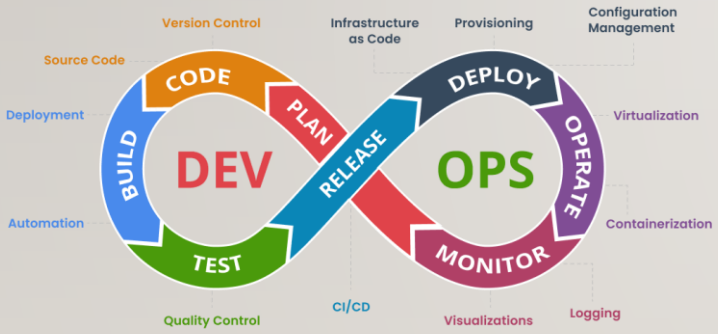
INTRODUCTION TO DEVOPS



עקרונות DevOps

- **ניטור ולוגים (Monitoring & Logging)**. ניטור ולוגים הם כלים חיוניים לניהול ותחזוקה של מערכות מידע. הם מאפשרים:
  - **מעקב אחר ביצועי המערכת:** ניטור מאפשר לעקוב אחר מדדים שונים של ביצועי המערכת (זמן תגובה, עומס, שגיאות) ולזהות בעיות בזמן אמת.
  - **איתור ותיקון בעיות:** לוגים מספקים מידע מפורט על פעילות המערכת ומאפשרים לאתר ולתקן בעיות במהירות.
  - **שיפור ביצועים:** ניתוח לוגים ונתוני ניטור יכול לסייע באופטימיזציה של ביצועי המערכת.

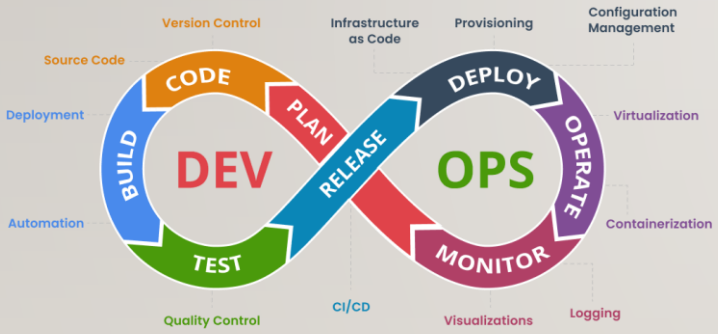
INTRODUCTION TO DEVOPS



עקרונות DevOps

- **עבודה עם Microservices**. Microservices היא ארכיטקטורה של מערכות מידע המבוססת על פירוק המערכת לשירותים קטנים ובלתי תלויים. עבודה עם Microservices מאפשרת:
  - **גמישות ו-Scalability:** כל שירות יכול להיפרס ולהתעדכן בנפרד, מה שמאפשר גמישות רבה יותר ו-Scalability.
  - **פיתוח מהיר יותר:** צוותים שונים יכולים לעבוד על שירותים שונים במקביל, מה שמאיץ את קצב הפיתוח.
  - **עמידות גבוהה יותר:** אם שירות אחד נופל, שאר השירותים ממשיכים לעבוד.

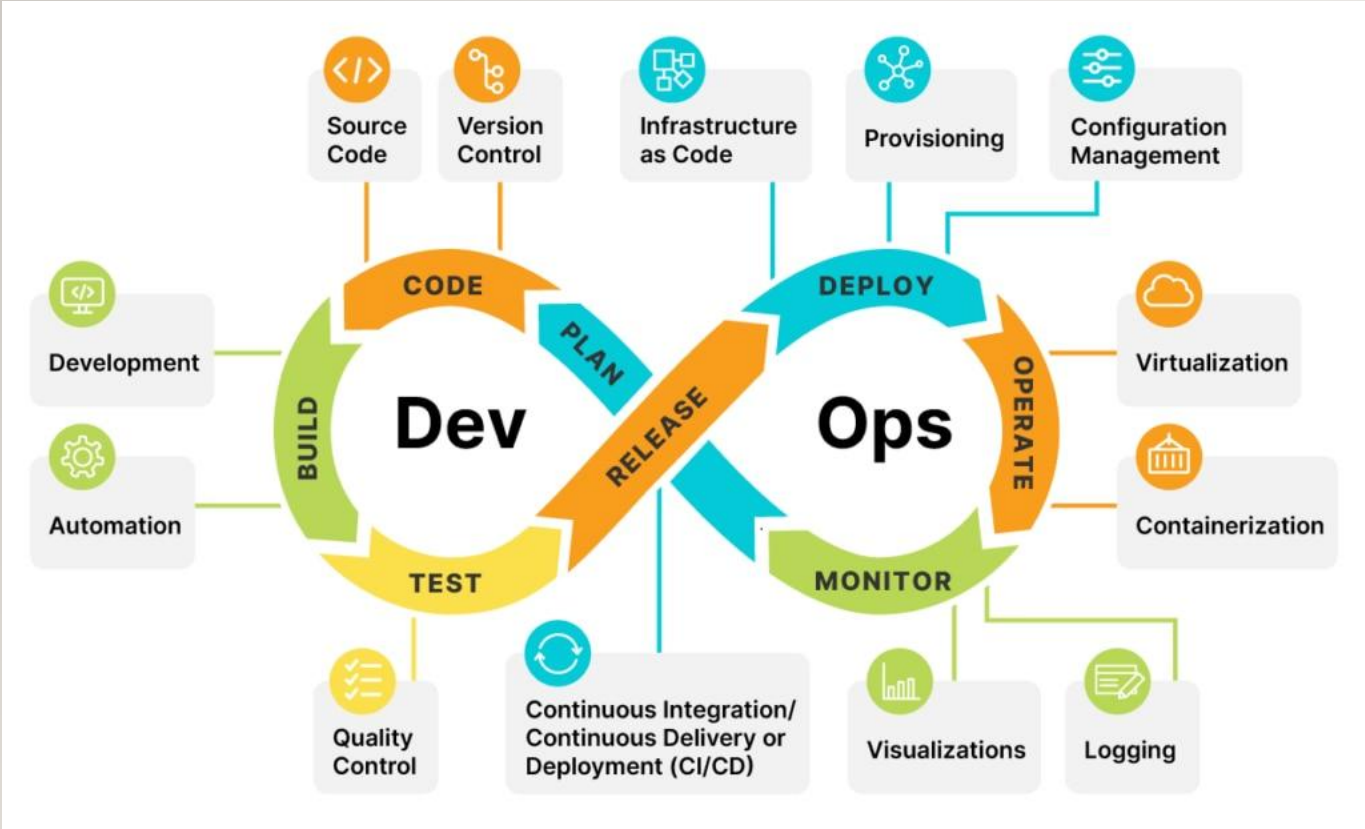
INTRODUCTION TO DEVOPS



עקרונות DevOps

- **תרבות שיתוף פעולה בין צוותים**. DevOps מדגישה את חשיבות שיתוף הפעולה והתקשורת בין צוותי הפיתוח והתפעול. תרבות שיתוף פעולה טובה מאפשרת:
  - **הבנה טובה יותר של הצרכים:** צוותים שעובדים יחד מבינים טוב יותר את הצרכים והאתגרים של כל צוות.
  - **פתרונות יעילים יותר:** שיתוף פעולה מאפשר יצירת פתרונות יעילים יותר לבעיות.
  - **שיפור מורל העובדים:** עבודה בצוות מגובש ותומך משפרת את מורל העובדים ואת שביעות הרצון שלהם.

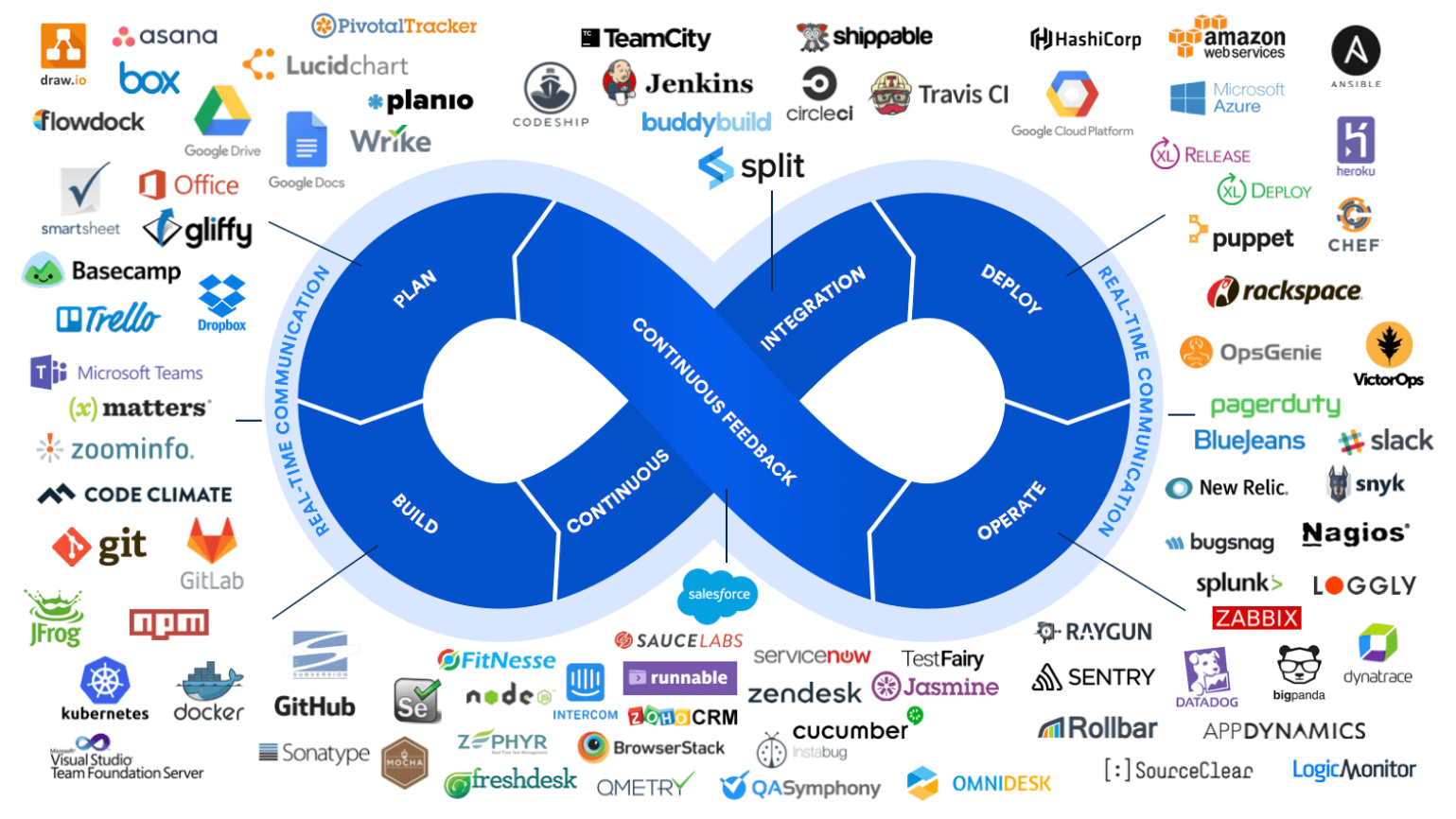
INTRODUCTION TO DEVOPS

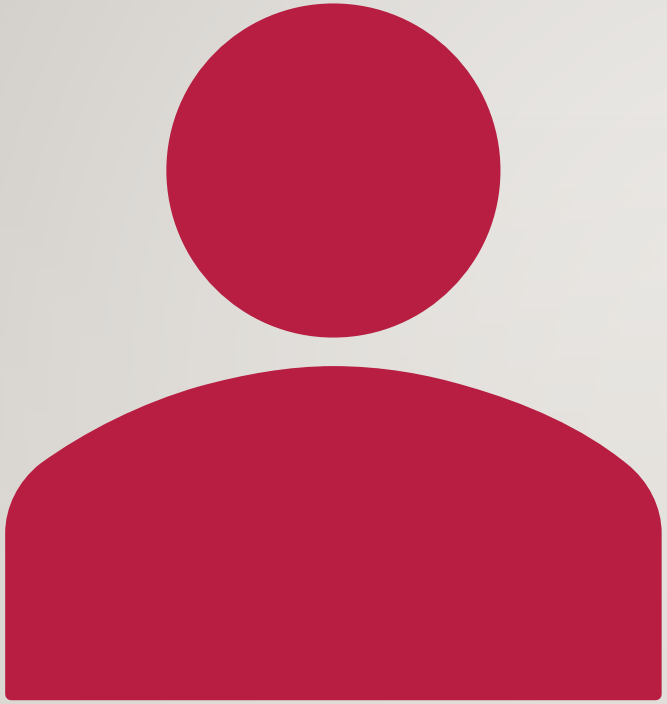


מחזור חיי הפיתוח Development Lifecycle

- תכנון: כתיבת קוד
- פיתוח, כולל ניהול גרסאות (GitHub).
- בדיקות, כולל בדיקות אוטומטיות (CI/CD).
- שחרור: פריסה בסביבת ייצור.
- ניטור ושיפור מתמיד.

INTRODUCTION TO DEVOPS





CLOUD NATIVE FUNDAMENTALS

CLOUD NATIVE FUNDAMENTALS

מה זה Cloud Native?

גישה חדשנית לפיתוח אפליקציות הממקסמת את הענן. במקום "להרים ולהעביר" אפליקציות מסורתיות לענן, Cloud Native משתמשת בכלים ובטכנולוגיות חדשניות כדי לבנות אפליקציות שתוכננו מלכתחילה לענן.

דוגמאות:

- כמו אפליקציות בנקאיות - בעבר הייתם צריכים ללכת לסניף, היום הכול מתבצע בענן בזמן אמת מכל מכשיר.
- נטפליקס לא קונה שרתים, אלא משתמשת בענן כדי להתמודד עם מיליוני צופים במקביל.

CLOUD NATIVE FUNDAMENTALS

מה זה Cloud Native?

מרכיבי המפתח:

- **מיקרו-סרוויסים (Microservices):** שירותים קטנים ובלתי תלויים. גמישות ומהירות.
- **קונטיינרים (Containers):** אריזה של אפליקציה וכל התלויות שלה. עקביות בין סביבות.
- **תשתיות אוטומטיות:** ניהול אוטומטי של תשתיות ענן. פריסה מהירה וחסכון.
- **סקלאביליות דינמית (Scalability):** התאמה אוטומטית של משאבים לעומס. זמינות גבוהה.

דוגמה:

באפליקציית משלוחים, מערכת התשלום, הניווט וההזמנות הן Microservices שונים שמדברים ביניהם דרך API.

CLOUD NATIVE FUNDAMENTALS

The CNCF Landscape

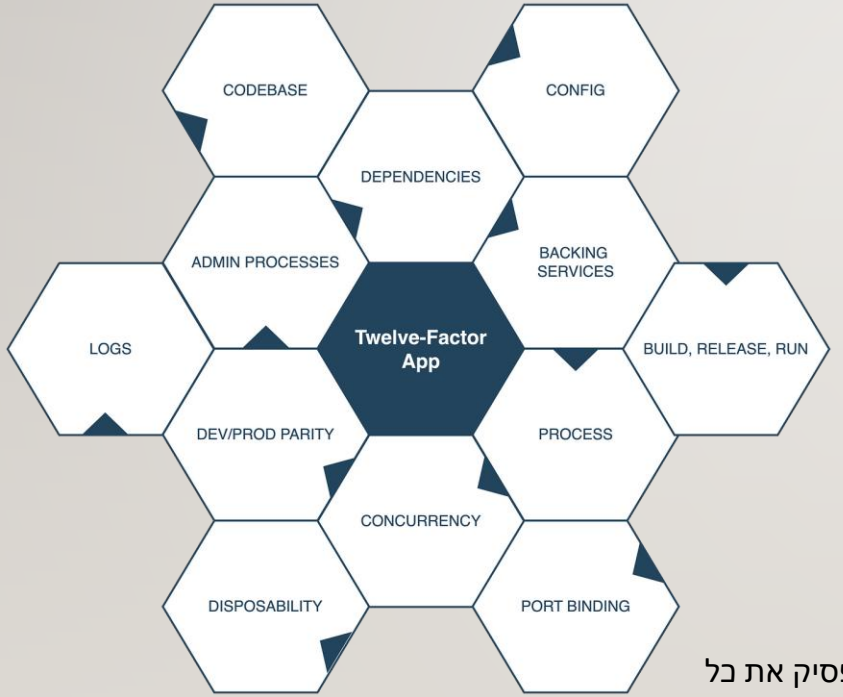
CNCF = Cloud Native Computing Foundation - הוא ארגון גג שתפקידו לטפח ולקדם טכנולוגיות Cloud Native. הוא ביתם של פרויקטים רבים וחשובים, כמו Kubernetes ועוד.

ה-CNCF Landscape הוא כלי ויזואלי שמציג את מגוון הכלים והטכנולוגיות בעולם ה-Cloud Native.

https://landscape.cncf.io/

CLOUD NATIVE
COMPUTING FOUNDATION

CLOUD NATIVE FUNDAMENTALS



Twelve-Factor App

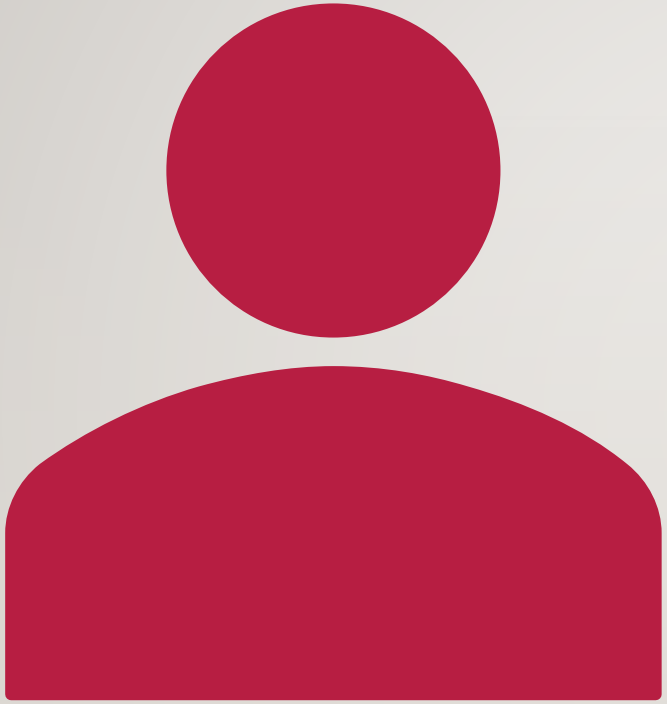
זו גישה שמגדירה איך לפתח אפליקציות מבוססות ענן בצורה נכונה.

עקרונות מרכזיים:

- ניהול קוד במאגר אחד (Git).
- הפרדת קונפיגורציות מסביבה לקובץ ייעודי.
- תהליכי בנייה, שחרור והרצה מובחנים.
- ניטור וניהול לוגים מובנים.

דוגמה:

כמו מערכת ייצור שבבים - כל שלב בייצור הוא עצמאי, ניתן לשפר או להחליף שלב מסוים מבלי להפסיק את כל קו הייצור.



DOCKER FUNDAMENTALS

DOCKER FUNDAMENTALS

Docker - עקרונות בסיסיים

Docker מאפשר להריץ קוד בצורה מבודדת בקונטיינרים שמכילים את כל מה שהאפליקציה צריכה כדי לרוץ.

דוגמה:

נניח שאתם מפתחים כלי חדש לבדיקות שבבים. הכלי תלוי בספריות ספציפיות ובגרסאות מסוימות של מערכת ההפעלה. Docker מאפשר לכם ליצור סביבה מבודדת שכוללת את כל מה שהכלי צריך, מבלי להסתמך על התקנות מקומיות או חשש להתנגשויות עם כלים אחרים. זה מבטיח שהכלי שלכם יעבוד בצורה עקבית בכל סביבה, ויקל על שיתוף פעולה עם צוותים אחרים.

DOCKER FUNDAMENTALS

מה זה קונטיינר?

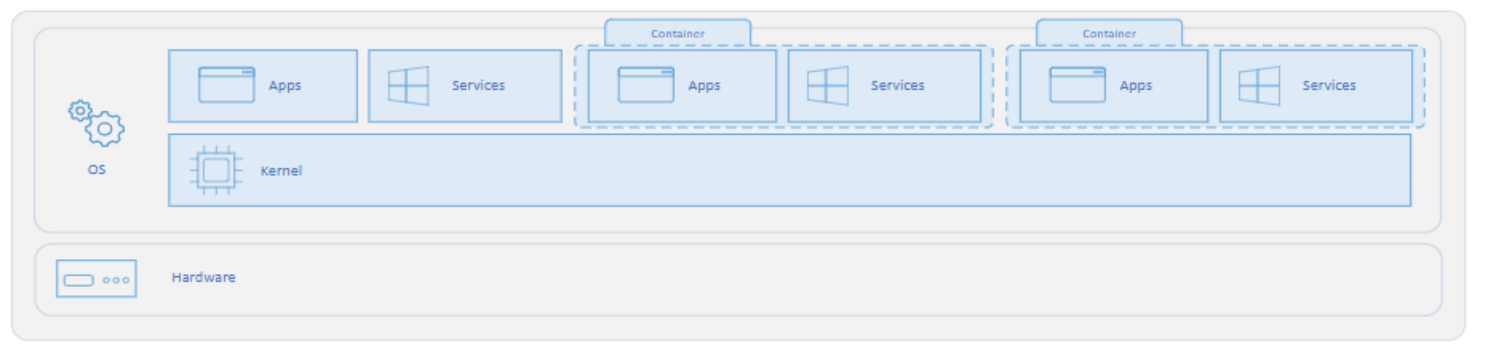
קונטיינר הוא יחידת תוכנה עצמאית שכוללת את כל מה שהאפליקציה צריכה כדי לרוץ: קוד, ספריות, הגדרות וכל התלויות האחרות.

למה להשתמש ב-Docker?

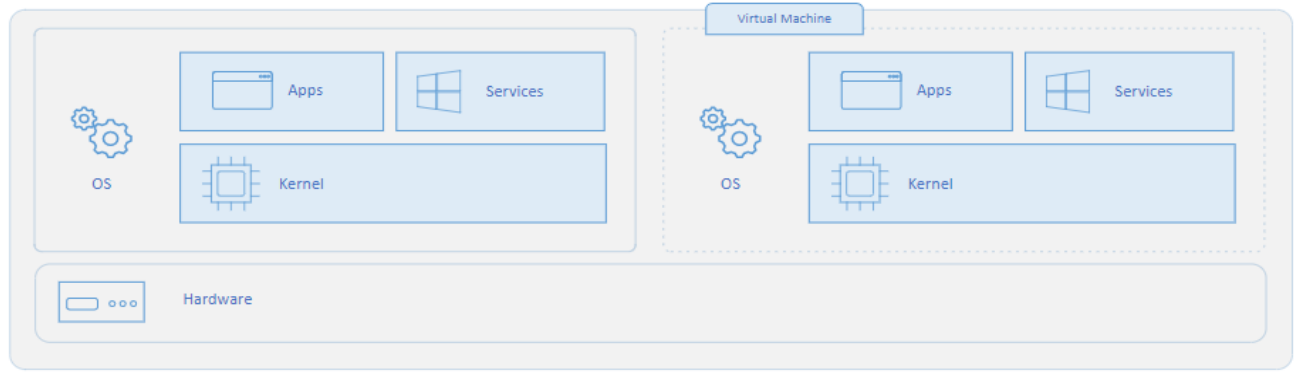
- **בידוד**: קונטיינרים מבודדים זה מזה ומהמערכת ההפעלה המארחת, מה שמבטיח שאפליקציה אחת לא תשפיע על אחרת.
- **ניידות**: ניתן להפעיל קונטיינר בכל סביבה: מחשב פיתוח, שרת בענן או שרת מקומי.
- **יעילות**: קונטיינרים קלים יותר ממכונות וירטואליות ודורשים פחות משאבים.
- **פיתוח מהיר**: Docker מאפשר פיתוח ופריסה מהירים יותר של אפליקציות.

DOCKER FUNDAMENTALS

Container



VM



VM vs Container

VM (מכונה וירטואלית):

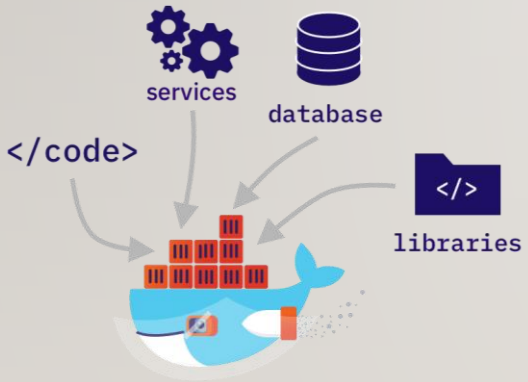
מחשב שלם בתוך מחשב.

כבד, איטי.

קונטיינר: אפליקציה ארוזה עם כל מה שהיא צריכה.

קל, מהיר.

DOCKER FUNDAMENTALS



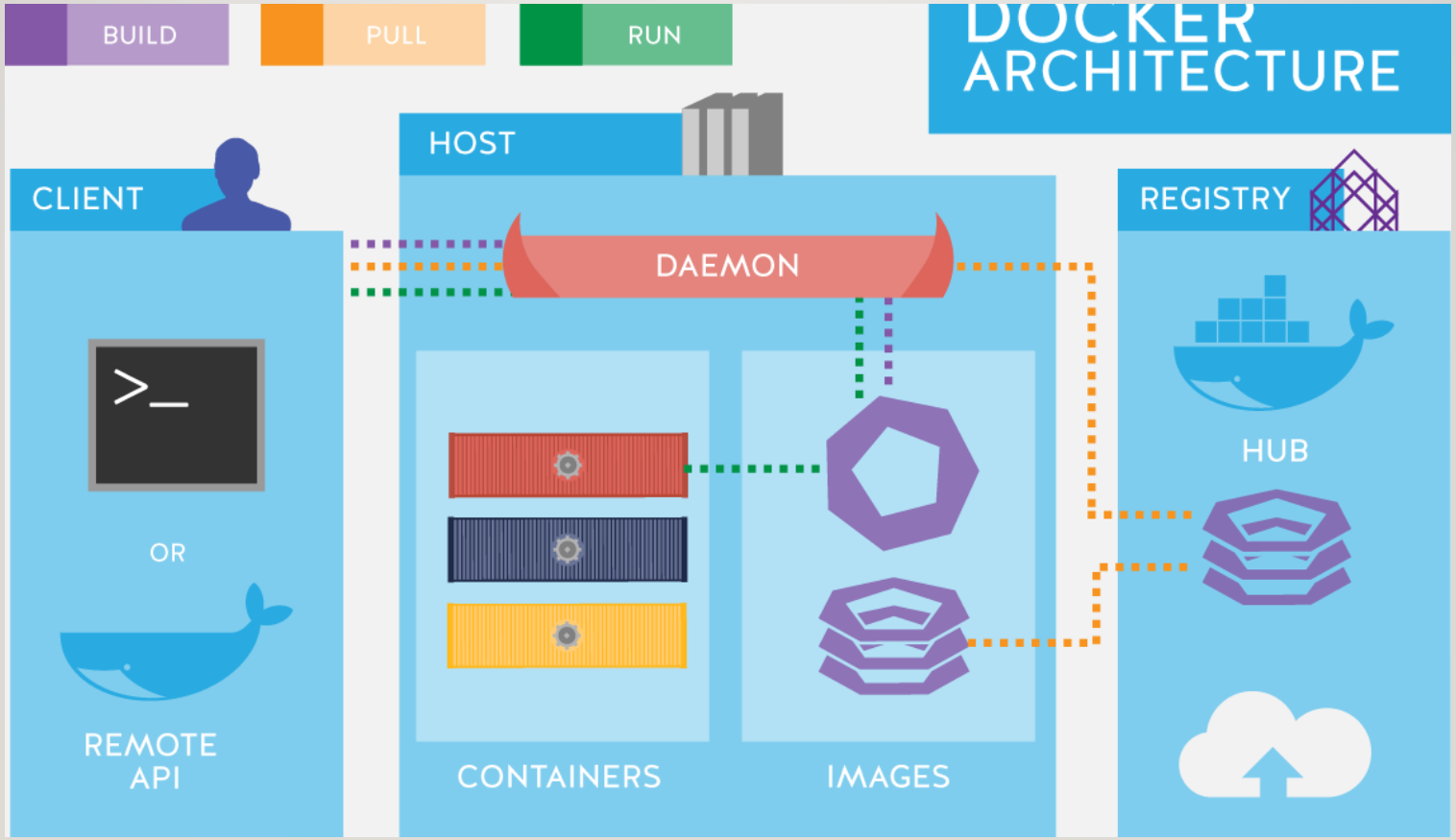
ארכיטקטורת Docker

Docker מורכב ממנוע הרצה, תמונות מוכנות מראש, קונטיינרים שרצים בפועל ו-Docker Hub שמאפשר שיתוף קבצים. כמו הורדת אפליקציות מחנות - אתם מורידים תמונה מוכנה, והיא רצה כמו שצריך בלי התקנות נוספות.

המרכיבים הם:

- Docker Engine
- Docker Images
- Docker Containers
- Docker Hub

DOCKER FUNDAMENTALS



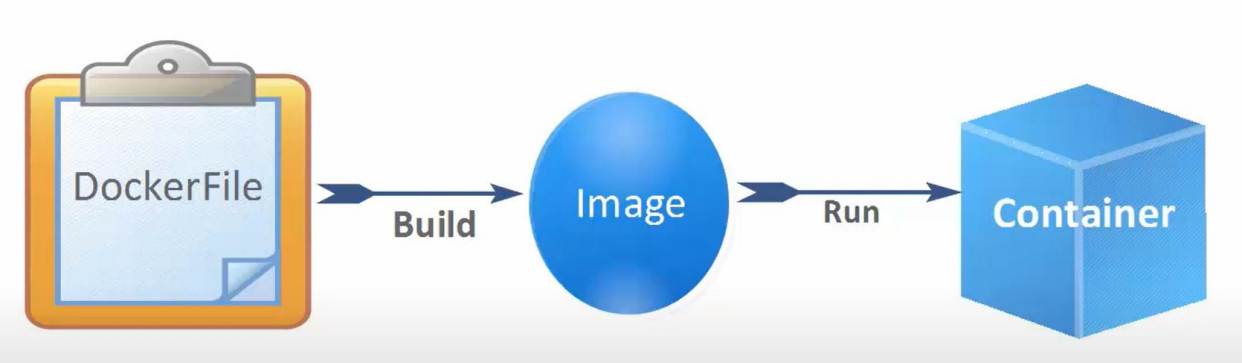
DOCKER FUNDAMENTALS

פקודות בסיסיות ב-Docker CLI

הפקודות מאפשרות לנהל קונטיינרים בקלות - להוריד, להריץ, לעצור ולבדוק אותם.

הפקודות הבסיסיות הן:

- docker pull
- docker run
- docker ps
- docker images
- docker build
- docker push



DOCKER FUNDAMENTALS

יצירת Dockerfile

Dockerfile מגדיר איך לבנות קונטיינר - איזה בסיס לקחת, איזה ספריות להוסיף ואיך להגדיר את הסביבה.


```
FROM python:3.7

RUN mkdir /app
WORKDIR /app
ADD . /app/
RUN pip install -r requirements.txt

EXPOSE 8080
CMD ["python", "/app/main.py"]
```

DOCKER FUNDAMENTALS

דוגמה

Docker commands:

- docker build -t app-name:latest .
- docker images
- docker ps -a
- docker run --rm -p 5000:5000 app-name:latest
- docker tag far-2-cel:latest agorbach/far-2-cel:latest
- docker push agorbach/far-2-cel:latest

שאלות?