# **Docker**

**OS – Operating System**

אותה תוכנה שמותקנת כדבר ראשון במחשב ומקשרת בין התוכנות השונות לחומרת המחשב.

שלושת הגדולים: Windows, Linux, Mac OS

**מכונה וירטואלית**

תוכנה המריצה בתוכה מערכת הפעלה שלמה בתוך מחשב שמכיל מערכת הפעלה.

מחשב נקרא מכונה או Host

מכונה וירטואלית נמצאת כתוכנה בתוך מחשב המכיל מערכת הפעלה ראשית.

לכל מערכת הפעלה – ראשית או כמכונה וירטואלית יש

1. מערכת קבצים ותיקיות משלה.
2. רשימת Ports עבור תקשורת מחוץ למכונה (הפיזית או הוירטואלית)

מכונה וירטואלית הינה כבדה מאוד. היא תופסת נפח רב, תופסת המון זיכרון, תופסת הרבה CPU.

מכונה וירטואלית כוללת את כל הנפח של מערכת ההפעלה.

**Docker**

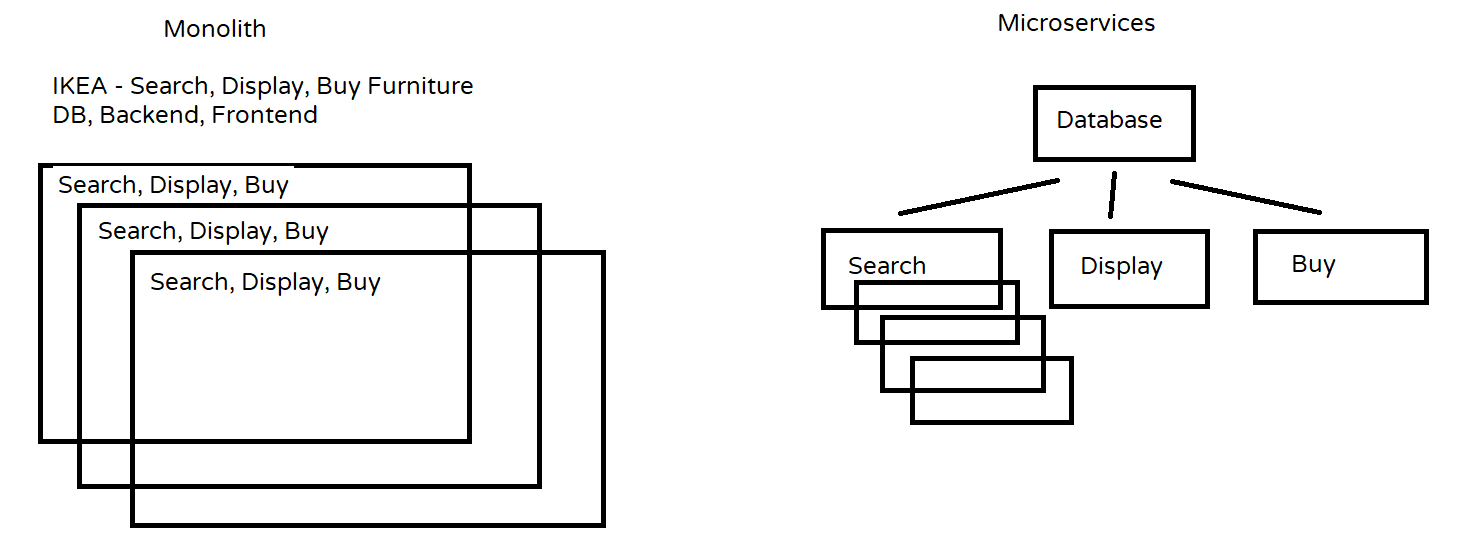
מערכת המאפשרת לבנות את האפליקציה שלנו בתוך Container המהווה מעיין "מכונה וירטואלית" במקום בניית האפליקציה ישירות בתוך מחשב הפיתוח.

זה מאפשר לבצע את ההתקנות הדרושות לאפליקציה בתוך ה-Container במקום במחשב עצמו.

זה מאפשר לאחר מכן להריץ את Container במחשב אחר.

זה מאפשר להריץ מספר Containers שכל אחד מכיל את האפליקציה שלנו

מאפשר לבנות את האפליקציה כ-Microservices במקום Monolith ואז לשכפל Microservices שמכילים לחץ עליהם



**Docker Image**

אלו כלל הקבצים הדרושים עבור ה"מכונה הוירטואלית" או ה-Container.

זוהי מעיין רשימת מכולת – מה דרוש להתקין מאפס במחשב חדש בכדי להריץ את האפליקציה שלנו.

Docker Image כולל:

1. מערכת הפעלה (Windows או Linux או Mac OS)
2. התוכנה עליה רצה האפליקציה שלנו (Node.js או .NET או Java או Python וכו')
3. כל ספריית 3rd Party הדרושה לאפליקציה שלנו
4. הקבצים של האפליקציה שלנו

**Container**

זהו Process (תוכנה שרצה במחשב) שמהווה מכונה וירטואלית מאוד מצומצמת מבחינת נפח, דיסק, CPU וכו'.

Container משתמש ב-Kernel של מערכת ההפעלה של מחשב ה-Host (Kernel זוהי הליבה של מערכת ההפעלה).

**Dockerfile**

זהו קובץ בעל השם Dockerfile (Case-Sensitive, ללא סיומת) המכיל את אותה "רשימת קניות" של אלו מע' הפעלה/קבצים/תוכונות וכו' צריך להיות במחשב חדש מהניילונים בכדי להריץ את האפליקציה שלנו.

הוא מגדיר מה יש ב-Docker Image.

כל Image בנויה משכבות שכבות. כל פקודה ב-Dockerfile הינה שכבה חדשה. Docker מצבע Cache על כל שכבה שאנו מורידים. אם אנו בונים Image והשכבה קיימת ב-Cache – היא תילקח מה-Cache ו-Docker לא יצטרך להוריד אותה. אם השכבה שונתה מהפעם האחרונה – Docker יוריד אותה ואת כל השכבות שלאחריה, אפילו שהם לא שונו.

**Linux**

ליבה של מערכת הפעלה, Open Source, מאוד מהירה.

ניתן לקחת את הליבה הזו, לבצע עבורה Fork (גרסה אחרת) ולפתח מערכת הפעלה נוספת על בסיס Linux.

זה נקרא Linux Distribution. יש מעל 1000 כאלו.

לדוגמה: Debian, Alpine, Red Hat, Android, Ubuntu.

**Port Binding**

קישור בין תקשורת המגיעה ל-Port ב-Host לתוך Port ב-Container.

פקודות טרמינל

בניית Image:

docker build -t <image-name> <Dockerfile-location>

לדוגמה:

docker build -t cute-kittens-backend .

בכדי שניתן יהיה להעלות את ה-Image ל-Repository (כמו Docker-Hub) על שם התמונה להתחיל ב-ID שלכם השייך ל-Repository. לדוגמה:

docker build -t bartsimpson/cute-kittens .

חשוב גם לתת גרסה לשם התמונה:

docker build -t bartsimpson/cute-kittens:1.0 .

יצירת Container:

docker create --name <container-name> -p <host-port>:<container-port> <image-name>

לדוגמה:

docker create --name backend-container -p 3002:3001 bartsimpson/cute-kittens:1.0

**Docker Compose**

זהו קובץ המכיל הגדרות ליצירת התמונה הסופית מקובץ ה-Dockerfile, יצירת Container ועוד.

בנוסף הוא יכול להכיל הגדרות של מספר Containers.

זה מאפשר לחסוך את פקודות הטרמינל הרבות שהגדרנו.

סיומת הקובץ הינה yaml או yml שזו שפה שדומה בתפקידה ל-JSON או XML בכדי לבנות קובץ הגדרות/קונפיגורציה/מידע וכו'.

YAML: Yaml Aint Markup Language

בנייה והרצת Docker Compose:

docker-compose up

docker-compose up -d

-d מציין הרצה במצב Detached, כלומר הטרמינל לא נתפס ע"י ה-Container אלא פנוי להמשך שימוש.

**Docker Volume**

זה קישור בין תיקייה שנמצאת ב-Host לתיקייה שנמצאת ב-Container.

כל קובץ שמוסיפים/מעדכנים/מוחקים מתיקיית ה-Host יתווסף/יתעדכן/ימחק אוטומטית מתיקיית ה-Container וההיפך.

ברגע ש-Container נכבה, כל ה-File System שלו נעלם/נהרס. אם האפליקציה שלנו יצרה קובץ – הוא ימחק.

לכן אם יש לו Volume – הוא ישמר גם בתיקיית ה-Host.

כאשר ה-Container מופעל שוב – כל הקבצים שבתיקיית ה-Volume של ה-Host מוחזרים בחזרה לתיקיית ה-Volume של ה-Container.

זה מאפשר לנו לבצע Persist Data. לדוגמה טיפול ב-DB.

קיימים שלושה סוגי Volumes:

1. Host-Volume

קישור תיקייה ספציפית ב-Host לתיקייה ספציפית ב-Container

1. Anonymous Volume

קישור תיקייה אנונימית ב-Host לתיקייה ספציפית ב-Container. שם התיקייה שב-Host יהיה שם רנדומלי.

בכל הרצה של Container שם התיקייה ב-Host יהיה שם אנונימי חדש.

1. Named Volume

קישור תיקייה אנונימית ב-Host המכילה מבחינתנו Friendly Name לתיקייה ספציפית ב-Container.

בכל הרצה של Container התיקייה ב-Host תחזור להיות אותה התיקייה.

בסוף Docker Compose יש לבצע רשימה של כל ה-Named Volumes

במערכת הפעלה Windows מיקום Anonymous Volume ו-Named Volume הינו:

[\\wsl$\docker-desktop-data\version-pack-data\community\docker\volumes](file:///\\wsl$\docker-desktop-data\version-pack-data\community\docker\volumes)