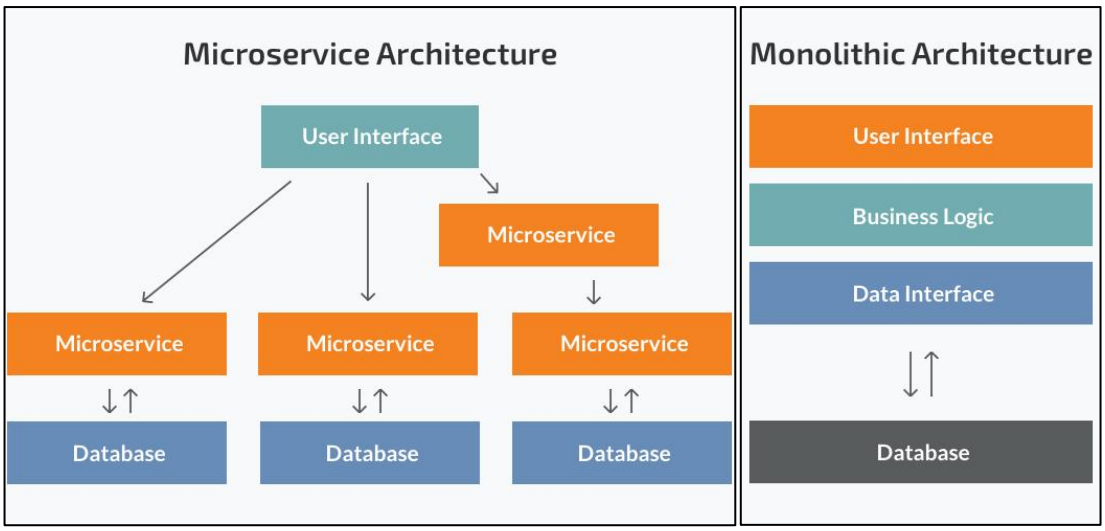
22.2.22

הנדסת פתרונות תוכנה מתקדמים- המשך מבוא

- Stack Overflow: 3 כללים ליצירת שאלות
  1. Problem statement- פירוט הבעיה בפשטות.
  2. Sample code and data- דוגמת קוד שהיא stand-alone ולציין מה הקוד צריך לעשות.
  3. Spelling, grammar and formatting- ניסוח באנגלית תקינה וברורה.
- חלוקת הציון:
  1. 25% מהציון- ביצוע מודלים בAWS על הנושאים: S3, EC2, EBS, RDS. בונוס 10 נקודות למי שמסיים את כל ההכשרה. **תאריך סיום- 30/04/22**
  2. 25% מהציון- בניית צד שרת backend באמצעות fastapi +REST/HTTP Dockerization. **תאריך סיום- 01/04/22**
  3. UI באמצעות react/streamlit. **תאריך סיום- 01/05/22**
  4. Docker compose לכל מה שנבנה בסעיפים 3+2 וכתיבת README עם git submodules. **תאריך סיום- 29/05/22**
  5. פרזנטציה של המערכת- רצוי ליצור סרטון ביוטיוב של 2-3 דקות עם כיתוב באנגלית תקינה, בנוסף חייב README. **תאריך סיום- 29/05/22**
- תוכנות שחשוב להוריד:
  - ✓ WSL- נשלח מדריך בדיסקורד.
  - ✓ Windows Terminal- דרך חנות Microsoft.
  - ✓ Docker Desktop- לחפש בגוגל "docker download".
  - ✓ Visual Studio Code- מהאינטרנט.
- עקרונות חשובים בהנדסת תוכנה:
  - ❖ Measure twice and cut once- תכנון מקדים לפני ביצוע.
  - ❖ Don't Repeat Yourself (DRY)- להמנע מחזרות של קוד.
  - ❖ Keep It Simple -Stupid- (KISS)- פשטות.
  - ❖ You Aren't Gonna Need It (YAGNI)- לא לבצע over engineering, אין צורך לממש הכל בשלבים הראשונים של פיתוח התוכנה.
  - ❖ Avoid Premature Optimization- לא לבצע אופטימיזציה לפני שמקבלים תמונה כללית של התוכנה. (הקשר לDonald Knuth)
  - ❖ Principle Of Least Astonishment- לכתוב קוד מובן ואינטואיטיבי.
  - ❖ Law of Demeter- לבצע היררכיה נכונה בקוד, חלוקה של המחלקות והמודות לרפוסיטורים.
  - ❖ עקרונות SOLID.
- Monolithic vs. Microservices:
  - Monolithic– base code אחד לכל המערכות, יחידה אחת שמרכזת את כל ה-logic.
  - Microservices- ארכיטקטורה ששוברת כל אפלקציה ליחידות קטנות ועצמאיות.



- <u>Docker:</u> חברה שמספקת שירות שבו נלקח כלי (כמו מערכת הפעלה, סביבה של שפת תכנות וכו') בתצורת image ואותו "עוטפים" בcontainer שמריץ אותו.
האימג'ים יכולים להיות לוקליים (מותקנים על המחשב של המשתמש), ב-dockerhub או בשרת פרטי.
  - ❖ Image- קובץ של הכלי (tool) כמו מערכת הפעלה.
  - ❖ Container- התהליך של הרצת הimage.
  - ❖ Alias- שמות שונים לאותו הדבר (בקורס זה- לאותה גרסה של ubuntu יש כמה שמות שונים שנתייחס אליהם כ-tags).
- <u>פקודות של Docker (נבצע ב-WSL):</u>
  - Dockerhub/Registry : פנייה לענן שבו שמורים אימג'ים.
  - Dockerfile : מגדיר מה יהיה בdocker, מחליט איזה אימג' ייבנה (קונפיגורציה).
  - docker build : בונה image מה-dockerfile.
  - docker pull : המשתמש נותן לפקודה כלי (וגרסה) שהוא רוצה לעבוד עם האימג' שלו. במידה והאימג' לא קיים לוקלית מתבצעת הורדה של האימג' מ-dockerhub.
    דוגמה- docker pull ubuntu:20.04
    אם לא רושמים גרסה אז מתבצעת הורדה של הגרסה האחרונה latest.
  - docker run : יוצר container ל-image, מריץ את האימג'. אם האימג' לא קיים מבצע pull אוטומטית.
    אם מוסיפים לפקודה ti- זה הופך את הטרמינל לאינטרקטיבי (פותח שורה חדשה שבה ניתן לתת פקודות לcontainer).
    אם מוסיפים בסוף הפקודה /bin/sh מתבצעת הרצה בshell (command שונה).
    אם מוסיפים בסוף הפקודה /bun/top מוצג איזה services רצים.
  - docker ps : מציג את כל הcontainers שרצים כרגע.
  - docker images : מציג את כל הimages שקיימים לוקלית, כל גרסה מקבלת שורה נפרדת.
  - docker network ls : מייצר לכל מודול network בפני עצמו, ls מציג את סוגי הnetworks.
  - docker volumes : מאפשר לcontainer "לראות את העולם החיצוני" אליו. כלומר, נותן לו לגשת לקבצים מחוץ אליו. צריך בפקודה זו לרשום נתיב (path) למקום שאליו נרצה שהוא ייגש. (מבצע מיפוי של תיקייה חיצונית לתיקייה בתוך הcontainer, לדוגמה).
  - Docker kill : מוחק את הcontainer שהID שלו רשום בסוף הפקודה.

- <u>פקודות של Image (לאחר ששנכנסנו לסביבה, כלומר אחרי docker run):</u>
  - ls : יוצר רשימה של כל התיקיות (directories) שנמצאים במיקום הנוכחי.
  - ls -l : נותן פירוט לרשימה של התיקיות.
  - ps : מציג את כל התהליכים שרצים (processes)
- <u>פקודות נוספות:</u>
  - exit : חוזר חזרה למקום שהיינו בו לפני. לדוגמה, אם מבצעים יציאה מהאימג' נשלחים חזרה לWSL.
  - clear : ניקוי מסך.
  - wsl -l -v : מריצים פקודה זו ב-power shell (PS) והיא מציגה את התהליכים (כמו docker desktop, ubuntu) והסטטוס שלהם. (l זה list, v זה version).