# **Docker**

זו מערכת המאפשרת להריץ את כלל האפליקציה שלנו על מעיין "מכונה וירטואלית" הנקראת ב-Docker: Container.

מכונה – Machine – זהו המחשב עצמו – "ברזלים".

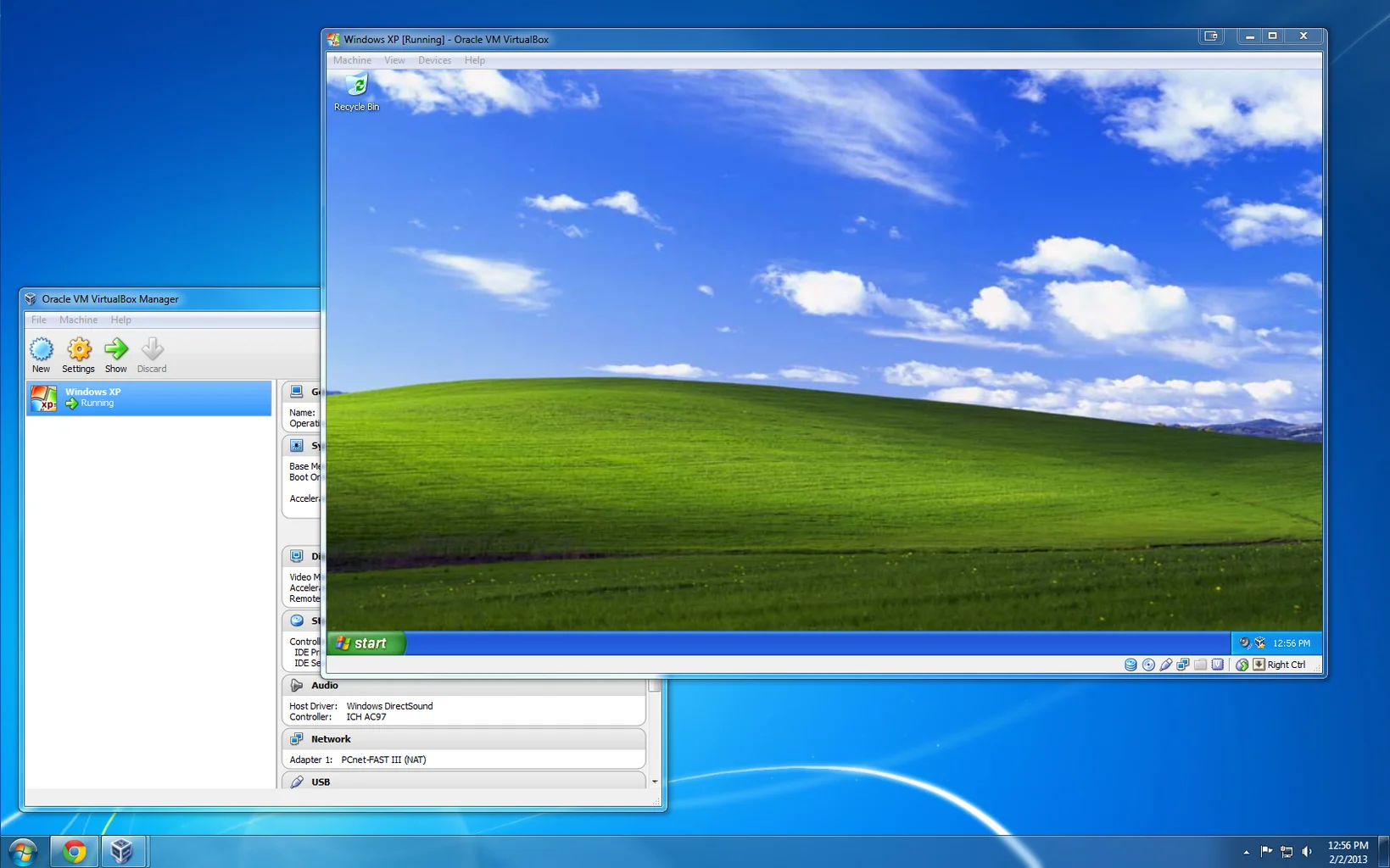
מערכת הפעלה – OS – Operating Machine: המערכת הראשונה המותקנת במחשב ומקשרת בין התוכנות לחומרת המחשב.

שלושת מערכות ההפעלה הגדולות:

1. Windows
2. Linux
3. Mac

מערכת הפעלה ראשית: מערכת ההפעלה שהותקנה על המכונה.

מכונה וירטואלית: Virtual Machine: התקנת מערכת הפעלה שלמה בתוך מערכת ההפעלה הראשית, ע"י אפליקציה המאפשרת זאת, לדוגמה Virtual Box.



מכונה וירטואלית זוללת המון זיכרון והמון מעבד.

Docker Container

מיני מכונה וירטואלית שמיועדת להריץ את האפליקציה שלנו.

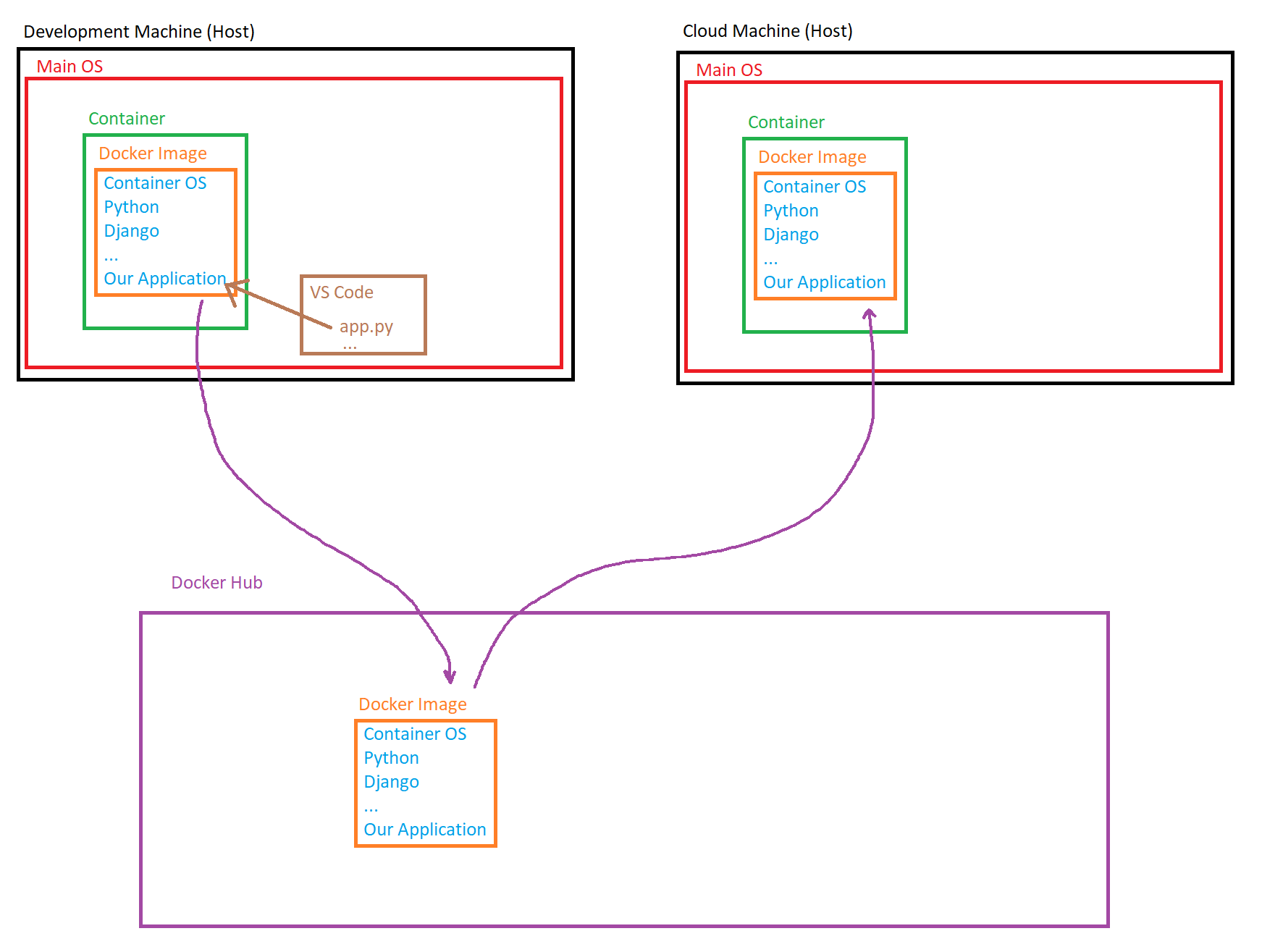
ב-Docker יש את התשתית של מערכת ההפעלה כך שה-Container יכיל מעט מאוד בתוכו כך שהוא לא יתפוס הרבה נפח.

Docker Image

זהו קובץ המכיל את כל מה שאמור להיות ב-Container בכדי להריץ בסופו של דבר את האפליקציה שלנו

במינימום חייבים להיות המרכיבים הבאים:

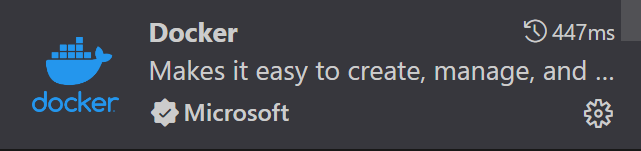
1. מערכת הפעלה.
2. סביבת הרצה של שפת התכנות, לדוגמה Python.
3. התקנות גלובליות הדרושות ע"י האפליקציה, לדוגמה Django.
4. האפליקציה שלנו.



Dockerfile

זהו קובץ בעל השם Dockerfile (מילה אחת, Case-Sensitive) המתאר איך לבנות Docker Image.

תוסף VS Code לעבודה עם Dockerfile (בודק שגיאות, מציע הצעות...)



פקודת טרמינל לבניית Docker Image:

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

לדוגמה:

docker build -t hello-python .

פקודת טרמינל לבניית Docker Container (ללא הרצתו):

docker create --name <container-name> <image-name>

לדוגמה:

docker create --name cool-container hello-python

פקודת טרמינל להרצת Docker Container:

docker start <container-name>

לדוגמה:

docker start cool-container

בכדי להעלות Docker Image ל-Docker Hub, על שם ה-Image להיות בפורמט הבא:

docker-username/image-name:version

לדוגמה:

bart-simpson/cute-kittens:1.0

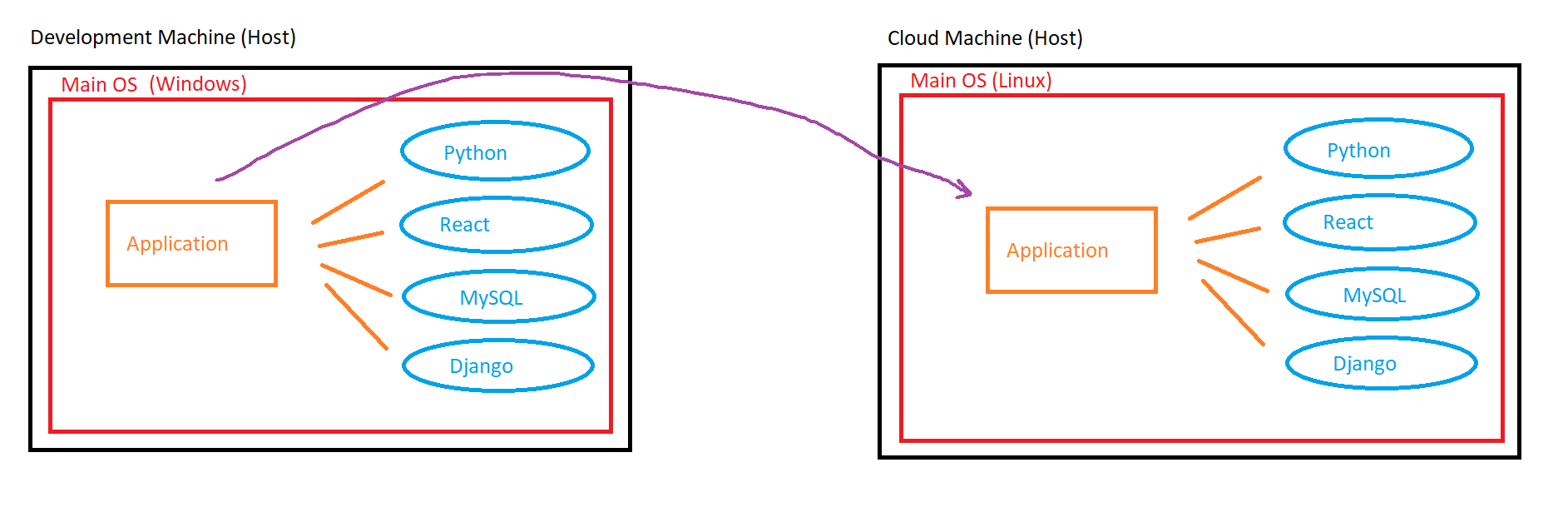
פקודת טרמינל להעלאת Docker Image ל-Docker Hub:

docker push <image-name>

לדוגמה:

docker push bart-simpson/cute-kittens:1.0

עבודה ללא Docker:



# 🐍 משתמשים בתמונה הרשמית של Python על בסיס Alpine Linux

# למה Alpine? → כי זו גרסה קלה מאוד, קטנה ומהירה להורדה (few MB בלבד).

# FROM מציין על איזה Image הבסיס שלנו נבנה.

FROM python:alpine3.19

# 📂 הגדרת תיקיית עבודה (Working Directory) בתוך הקונטיינר

# כל הפקודות הבאות (COPY, RUN, CMD וכו') ירוצו מתוך /app

# זה עוזר לנו לסדר את הקבצים במקום קבוע בתוך הקונטיינר.

WORKDIR /app

# 📥 מעתיקים את כל הקבצים מהמחשב המקומי (התיקייה הנוכחית עם Dockerfile)

# אל תוך התיקייה /app בתוך הקונטיינר.

# לדוגמה: אם יש לנו app.py במחשב, הוא יעבור ל־/app/app.py בקונטיינר.

COPY . /app

# 🚀 מגדירים את פקודת ההרצה (ENTRYPOINT) ברגע שהקונטיינר עולה.

# כאן אנחנו מריצים python3 עם הקובץ app.py

# כך שכל פעם שהקונטיינר יופעל → התוכנית הראשית שלנו תתחיל לרוץ.

ENTRYPOINT ["python3", "app.py"]

## 📌 סיכום והסברים עיקריים

1. **FROM** – בוחר תמונת בסיס (פה Python + Alpine).
2. **WORKDIR** – קובע את התיקייה הראשית שבה יעבוד הקונטיינר.
3. **COPY** – מעתיק קבצים מהפרויקט שלך לתוך הקונטיינר.
4. **ENTRYPOINT** – מגדיר מה הקונטיינר יריץ כברירת מחדל (כאן: python3 app.py).

**דוגמא מהתחלה עם דחיפה ל docker hub**

בונה לי image לא לשכוח - חובה שם המשתמש שלכם באתר docker hub לפני השם. סומן באדום

docker build -t asafamir/hello-python:1.0 .

דחיפת ה image ל docker :

docker push asafamir/hello-python:1.0

אם תרצו לבדוק לפני הדחיפה ל docker hub אז צרו קונטיינר על המחשב שלכם והריצו אותו

docker create --name cool-container asafamir/hello-python:1.0

docker start cool-container

אם תרצו למשוך את ה image שלכם או של משהו אחר תעשו זאת באמצעות פקודת pull

docker pull asafamir/hello-python:1.0

לאחר מכן תייצרו container תוכלו להריץ את האפפ שלכם

**הרצת דוקר עם env**

ניצור תיקיה src ובתוכה נשים את [app.py](http://app.py)

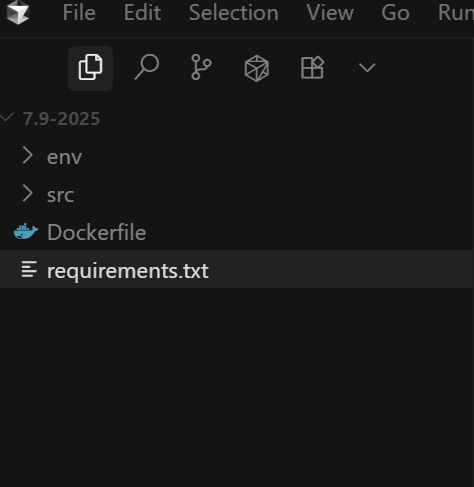
לאחר מכן ניצור תיקיית env :

python -m venv env

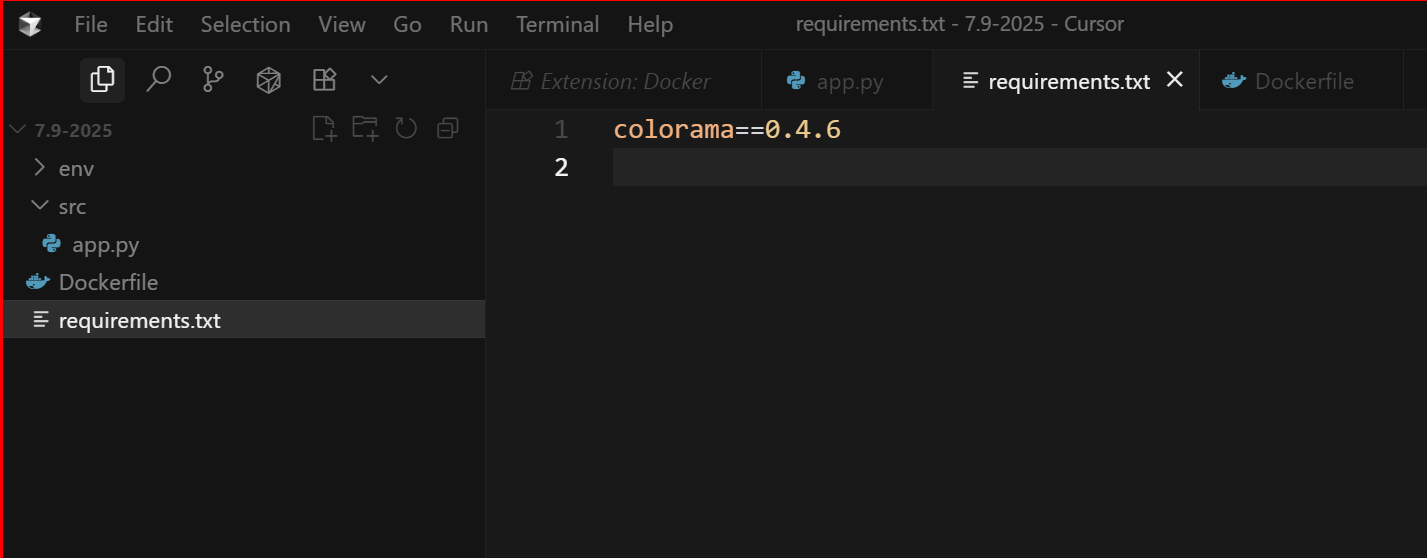
env/scripts/activate

pip install -r requirements.txt

ניצור קובץ requirements.txt



pip install colorama



pip freeze > requirements.txt

python src/[app.py](http://app.py)

from colorama import Fore

print(f"{Fore.GREEN}Hello World asaf")

print(f"{Fore.RED}Hello World asaf")

print(f"{Fore.YELLOW}Hello World asaf")

print(f"{Fore.BLUE}Hello World asaf")

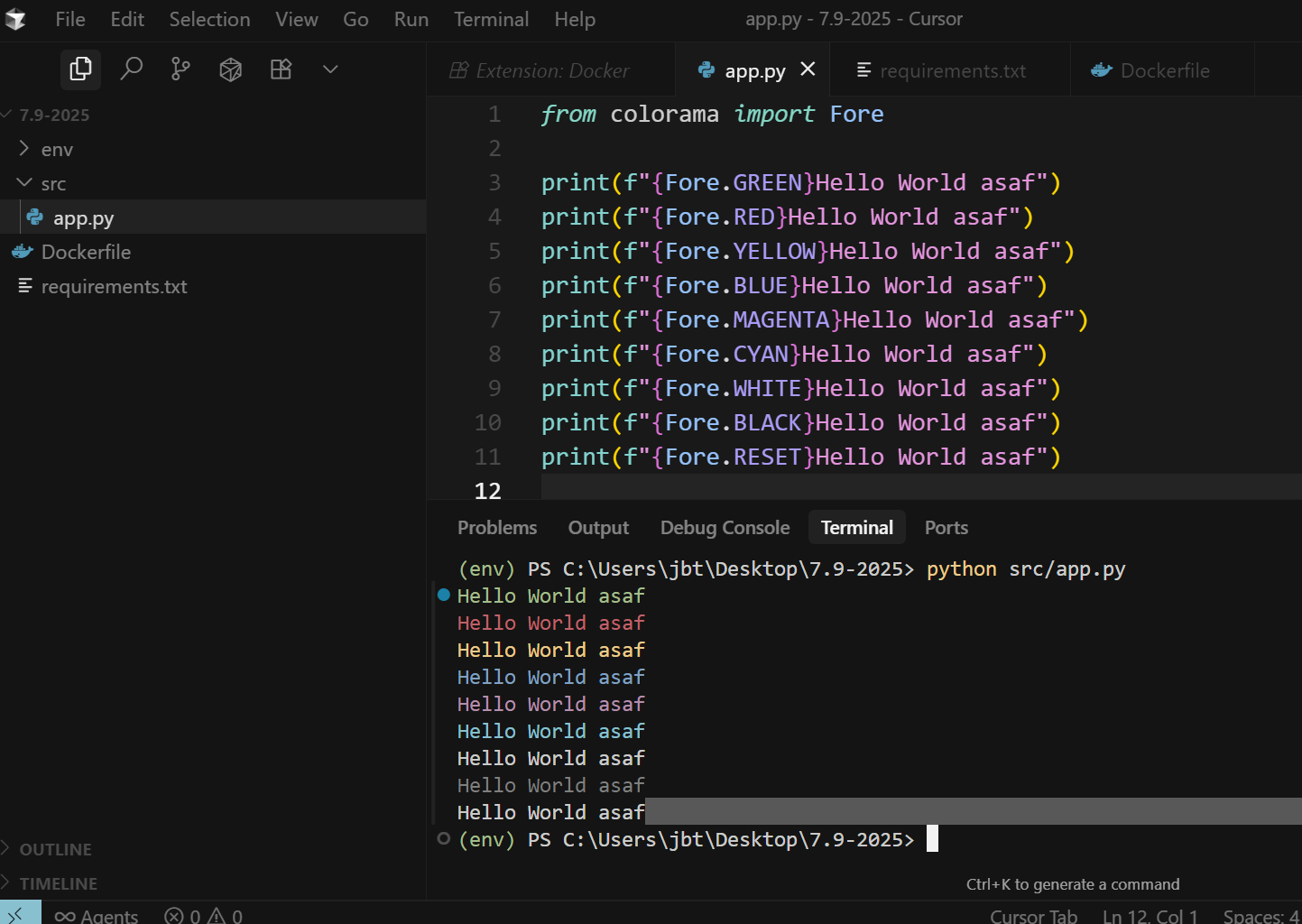
print(f"{Fore.MAGENTA}Hello World asaf")

print(f"{Fore.CYAN}Hello World asaf")

print(f"{Fore.WHITE}Hello World asaf")

print(f"{Fore.BLACK}Hello World asaf")

print(f"{Fore.RESET}Hello World asaf")



docker pull asafamir/colors-image:1.0

הסבר מפורט על הקוד

*#intall the linux alpine 3.19*

*#FROM alpine:3.19*

*#install the python3 and alpine 3.19*

FROM python:alpine3.19

*# 📂 הגדרת תיקיית עבודה (Working Directory) בתוך הקונטיינר*

*# כל הפקודות הבאות (COPY, RUN, CMD וכו') ירוצו מתוך /app*

*# זה עוזר לנו לסדר את הקבצים במקום קבוע בתוך הקונטיינר.*

WORKDIR */app*

*# 🧪 יוצרים סביבת פייתון מבודדת (virtualenv) בתוך /app בשם env*

RUN *python* *-m* *venv* *env*

*# 📄 מעתיקים את קובץ התלויות. שלב נפרד — משפר קאש של Docker*

COPY *requirements.txt* */app*

*# 📦 מתקינים את כל החבילות אל תוך ה־venv*

RUN */app/env/bin/pip* *install* *-r* *requirements.txt*

*# 📥 מעתיקים את כל הקבצים מהמחשב המקומי (התיקייה הנוכחית עם Dockerfile)*

*# אל תוך התיקייה /app בתוך הקונטיינר.*

*# לדוגמה: אם יש לנו app.py במחשב, הוא יעבור ל־/app/app.py בקונטיינר.*

COPY *.* */app*

*# 🚀 מריץ את האפליקציה דרך פרשן ה־Python של ה־venv*

ENTRYPOINT *[*"/app/env/bin/python"*,* "src/app.py"*]*