Using 1:

שלב 1:

הרצתי את הפקודה הבאה:



**הסבר:**

:docker run מפעיל קונטיינר חדש.

-d :מריץ את הקונטיינר ברקע (detached mode).

: -p ממפה את פורט 8181 במחשב המקומי לפורט 80 בקונטיינר, שם Nginx מאזין.  -: name my-nginx --: נותן שם לקונטיינר

: Nginx התמונה של Nginx מה-Docker Hub.-

שלב 2:

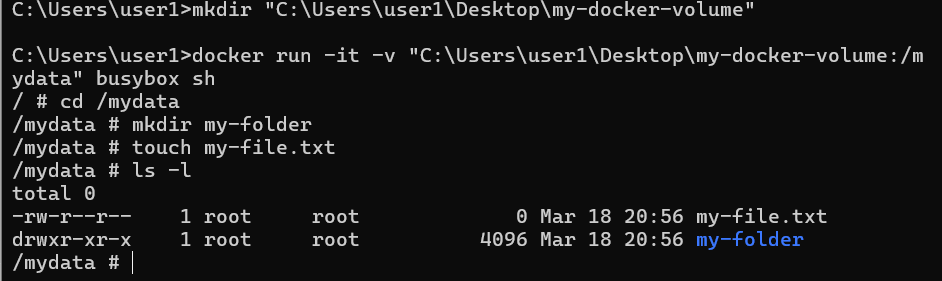
פתחתי בניתוב http://localhost:8181/ כדי לראות שאכן רץ וזה הדף שנפתח:

**The Purpose of the Exercises:**

Demonstrates how to expose a containerized application to the host machine. By mapping port 8181 on the host to port 80 in an Nginx container, the web server inside the container becomes accessible via http://localhost:8181. This is a fundamental concept in container networking.

Using 2:

הרצתי את הפקודות הבאות:



שלב 1: יצרתי תיקיה במחשב בשם my-docker-volume באמצעות הפקודה mkdir

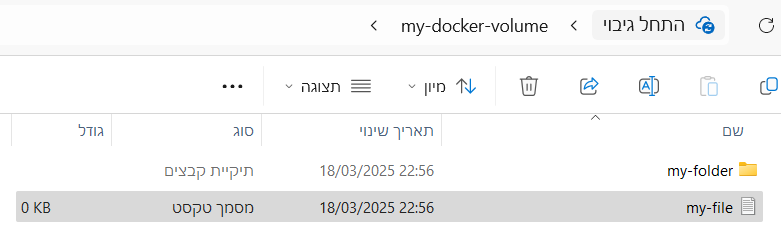
שלב 2: הרצת הפקודה:

it – : מאפשר עבודה אינטראקטיבית (פקודות ישירות).

-v ~/my-docker-volume:/mydata: ממפה את התיקייה my-docker-volume במחשב ל־/mydata בתוך

Busybox: התמונה של BusyBox (מערכת קטנה שמכילה פקודות בסיסיות)

Sh: מריץ את ה-Shell של BusyBox

שלב 3: בתוך הקונטיינר הרצתי את שאר הפקודות שיוצרות קבצים וכד...

**The Purpose of the Exercises:**

Explores how to share files between the host system and a container. By mapping a host directory to a BusyBox container, files created inside the container persist even after the container is removed. This technique is crucial for handling persistent data in Docker

Using 3:

הרצתי את הפקודה הבאה:



והתוצאה הבאה:



**The Purpose of the Exercises:**

Shows how to override a container’s default entrypoint using the --entrypoint flag. Instead of running the default Python process, the container starts a Bash shell and executes a command to check the installed Python version. This approach is useful for debugging and customizing container behavior.