מסמך אפיון - פרויקט NASA Asteroid Data Pipeline

1. מטרה

הפרויקט נועד להקים מערכת ששואבת נתונים מ-API של NASA לגבי אסטרואידים שנמצאים קרוב למסלול כדור הארץ. המערכת תשתמש בנתונים שנשלפים מה-API, תבצע עיבוד של המידע ותכניס אותו למסד נתונים מקומי.

הדרישות הן:

למשוך נתונים על אסטרואידים (שם, גודל, קירבה לכדור הארץ, מהירות).

לבצע עיבוד של הנתונים והכנסת המידע למסד נתונים.

הצגת 5 האסטרואידים הגדולים ביותר (לפי קוטר) שיתקרבו לכדור הארץ במהלך 30 הימים הקרובים.

מגבלות:

1. כמות מקסימום של טווח הימים בהם אפשר לבצע את הבקשה המקסימום הוא 7.
2. כמות מסוימת של בקשות שאפשר לבקש בטווח זמן קצר(OVER\_RATE\_LIMIT).

פתרון למגבלות:

1. יצירת לולאה שרצה 5 פעמים עם טווח של שישה ימים ככה הכמות ימים תהיה בדיוק 30.
2. שינוי חיבור לרשת אינטרנט ששולחת את הבקשה עם כתובת IP חיצונית שונה.

2. דרישות פונקציונליות

שאיבת נתונים: יש להתחבר ל-API של NASA ולשלוף את המידע על האסטרואידים.

פקודה הכוללת בקשה הנשלחת ל- API לקבלת המידע:



פתירת המגבלה של בקשה הכוללת 7 ימים מרחק מקסימום של מידע נפתרת בצורה הבאה:

הרצת לולאת FOR 5 פעמים, כל פעם מתבצעת הבקשה על מרחק של 6 ימים, ככה יוצא שהכמות ימים היא 5\*6 = 30 , בכל ריצה של הלולאה אחרי שהבקשה אל ה-API נשלחת והמידע שהוחזר הומר, יש קריאה לפונקציה שמכניסה את המידע אל המסד נתונים(database).

עיבוד נתונים: יש לעבד את המידע שנשלף מה-API ולהמיר אותו לפורמט שנדרש((JSON.

המרתו ל-JSON:



שליפת נתונים: יש להציג את 5 האסטרואידים הגדולים ביותר לפי קוטר, שיעשו קירבה לכדור הארץ ב-30 הימים הקרובים.

3. דרישות טכניות

שפות וטכנולוגיות:

Python

ספריות: requests, json, sqlite3, os, datetime

מסד נתונים: SQLite

מבנה הנתונים:

JSON – הפורמט שבו יגיע המידע מה-API

SQL – המידע ייאוחסן במסד נתונים SQLite.

4. שלבים לביצוע

הפרויקט מחולק לכמה שלבים, כל שלב יתבצע בנפרד:

התקנת ספריות - 10 דקות

התקנת הספריות הדרושות: requests, json, sqlite3, os, datetime.

התקנת הספריות מתבצעות על ידי הרצה באזורים המאפשרים כגון- CMD .

משיכת מידע מתוך ה-API והמרתו ל-JSON למשך 30 ימים קדימה - שעה

שליחת בקשה ל-API של NASA לקבלת מידע על אסטרואידים שיתקרבו לכדור הארץ ב-30 הימים הקרובים. המידע יוחזר בפורמט JSON.

יש להמיר את המידע ולשלוף את השדות הרלוונטיים: שם, קוטר, תאריך קירבה, מהירות.

הכנסת המידע למסד נתונים (SQLite) - חצי שעה

יצירת מסד נתונים SQLite וטעינת המידע שנשלף מה-API לתוך הטבלה המתאימה.

יישום פונקציה שמבצעת את הכנסת הנתונים למסד נתונים.

שליפת חמשת האסטרואידים הגדולים ביותר - 20 דקות

כתיבת קוד של SQL לשליפת חמשת האסטרואידים הגדולים ביותר (לפי קוטר) שיתקרבו לכדור הארץ ב-30 הימים הקרובים.

הצגת פרטי האסטרואידים על המסך.

5. הרחבות

מחיקת טבלאות תחילת כל ריצה.

המטרה של ההרחבה הזאת היא למנוע שכל פעם שתרוץ התוכנית יוכנסו ערכים כפולים למסד נתונים, יהיה ניתן לבצע זאת על ידי מחיקת הטבלאות של המסד נתונים.

אם אין מסד נתונים בשם שצוין בתוכנית באותה תיקיית הקוד יוצר מסד נתונים חדש.

6. בונוסים

6.1 הוספת טיפול בשגיאות (Error Handling)

בפרויקט הוספתי טיפול בשגיאות כדי להבטיח שהמערכת תוכל להתמודד עם בעיות שיכולות להתרחש בעת שליחה ל-API של NASA או בעיות בחיבור למסד הנתונים. טיפול בשגיאות הוא שלב חשוב על מנת להימנע מקריסת המערכת או חווית משתמש רעה במקרה של בעיות בלתי צפויות.

הדרישות לבונוס:

טיפול בשגיאות בעת שליחת בקשות ל-API:

הבקשה ל-API של NASA עשויה להיכשל עקב בעיות חיבור, קוד שגיאה לא 200 (למשל, אם יש בעיה במפתח ה-API או בשרת), או בעיות אחרות.

השתמשתי ב-requests.exceptions.RequestException כדי להתמודד עם בעיות רשת והגדרת קוד שגיאה שהיא לא 200.

גם ווידאתי שמתקבל תגובת JSON חוקית על ידי טיפול ב-ValueError במקרים בהם המידע לא ניתן להמרה כראוי.

6.2 הוספת DOCKER המכיל את ביצוע המשימה

בחרתי להשתמש ב-Docker בפרויקט כי הוא מאפשר לי להריץ את הקוד בסביבה מבודדת, כך שאני לא צריך לדאוג להתקנה של ספריות או תלויות בצורה ידנית בכל מחשב. Docker מבטיח שהקוד יעבוד בדיוק אותו דבר בכל מחשב, וזה מאוד חשוב כאשר עובדים על פרויקט צוות או רוצים להריץ אותו על מחשבים שונים. בנוסף, Docker מקל על הפצה ושיתוף של הסביבה עם אחרים, ומפשט את ניהול התלויות וההגדרות.

הדרך שבה אני יצרתי את הDocker היא על ידי שימוש בקובץ Dockerfile שהוא מריץ את הפקודות:

בניית התמונה:  
docker build -t nsotask .

הפקודה docker build בונה את התמונה (Image) על פי הקובץ Dockerfile שנמצא בתיקייה שבה הפקודה רצה.

התג -t : משמש לקביעת שם התמונה במקרה של הפרויקט הזה קבעתי את השם nsotask

(יש לשים לב שלא לשים אותיות גדולות בשם)

התג . (נקודה) – בסוף הפקודה מציין את התיקייה הנוכחית(שבה נמצא הDockerfile) שמדריך את הDocker כיצד לבנות את התמונה צריך להיות באותה תיקייה

בניית התמונה היא השלב שבו Docker לוקח את כל ההוראות שהגדרתי ב, Dockerfile-בונה את הסביבה, ומייצר תמונה שניתן להפעיל אותה כקונטיינר.

לשמור את הקובץ כקובץ TAR:

docker save -o nsotask.tar nsotask

הפקודה docker save שומרת את התמונה שהוזנה כקובץ TAR.

התג -:o מציין את שם הקובץ שאליו יישמר הimage. במקרה של הפרויקט הזה בחרתי nsotask.tar.

בסופו של דבר, הפקודה יוצרת קובץ TAR שמכיל את התמונה שנוצרה.

שמירת התמונה כקובץ TAR מאפשרת להעביר את התמונה למחשב אחר (למשל, במקרים של שיתוף או התקנה על מחשב אחר) מבלי צורך להתחבר לאינטרנט או לגשת ל- Docker Hub.

מדריך לטעינת ה-docker למחשב שנשלח אליו הקובץ TAR

לוודא שבמחשב מותקן TAR ו-docker.

למקרה ואין:

[התקנת docker.](https://www.docker.com/products/docker-desktop/)

[התקנה של TAR.](https://www.winzip.com/en/learn/file-formats/tar/?x-target=ppc&promo=ppc&utm_source=google&utm_medium=cpc&utm_campaign=wz-dd-all-adwordsppc&utm_content=&utm_term=&utm_id=19666049983&gad_source=1&gclid=Cj0KCQiA0MG5BhD1ARIsAEcZtwQTJSe3wMHfQcrc-xsIjZIDh762ljszUQdSZ3sQFnLDatZrNDLU-wAaAv-aEALw_wcB)

[התקנת python.](https://www.python.org/downloads/)

כניסה ל-TARMINAL

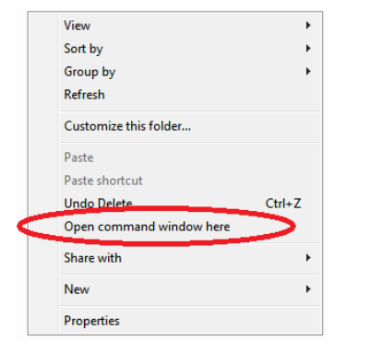
אחת הדרכים להריץ את ה-docker היא הטרמינל.

כניסה לטרמינל יכולה להתבצע בכמה דרכים:

Windows:

לגשת לתיקייה בה נשמר הקובץ, החזיקו את מקש ה-Shift ולחצו לחיצה ימנית בעכבר (לא על הקובץ, ולא כאשר הקובץ מסומן).

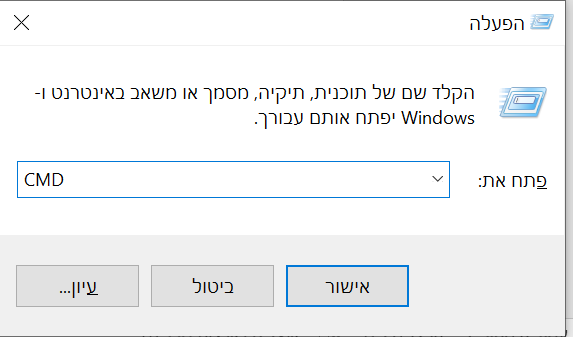
בתפריט שנפתח, לבחור "Open command window here" (בעברית: "פתח חלון פקודה כאן").



אפשרות נוספת היא על ידי כניסה לתיקייה בה שמרתם את הקובץ וללחוץ על אזור ה-path ואחרי לחיצה למחוק את מה שרשום ולהכניס CMD: 

ואז לחיצת המקש ENTER

אפשרות נוספת היא לחיצת על המקשים windows+r

יפתח המסך הזה:  לוודא שרשום בפנים CMD וללחוץ אישור ויפתח המסך טרמינל(cmd).

יש לשים לב שבשיטה הזאת הטרמינל נפתח לנו במיקום ברירת המחדל שלו ולא בגוף התיקייה דרך אחת לנווט אל מיקום התיקייה על ידי שימוש בפקודת cd.

אפשר לנווט בצורה הזאת: cd + path

שב-path יש את המסלול של המיקום קובץ

PowerShell:

לחיצה על מקש Windows, לחפש "PowerShell" ולחץ על Enter.

macOS

ב-macOS, ניתן לגשת לטרמינל בכמה דרכים:

דרך Finder:

לפתוח את Finder ולחץ על "Applications" בצד השמאלי.

כנס ל-"Utilities" ומצא את "Terminal".

באמצעות Spotlight Search:

ללחוץ על Command + Space כדי לפתוח את Spotlight.

לכתוב "Terminal" וללחוץ Enter.

גישה לתיקייה דרך הטרמינל:

ב-Finder, עבור לתיקייה הרצויה. לחץ עם מקש ימני ולבחור ב-"New Terminal at Folder" (זמין במערכות הפעלה חדשות יחסית).

Linux (למשל, Ubuntu, Fedora)

ברוב סוגי הלינוקס, ניתן לגשת לטרמינל כך:

דרך תפריט האפליקציות:

במערכת Ubuntu, ללחוץ על מקש Windows או על כפתור ה-"Show Applications" בפינה התחתונה של המסך, חפש "Terminal" ולחץ Enter.

קיצור מקלדת:

ברוב הסוגים של לינוקס, הקיצור לפתיחת טרמינל הוא Ctrl + Alt + T.

גישה לתיקייה דרך הטרמינל (Ubuntu):

פתח את תוכנת ניהול הקבצים.

לחץ על מקש ימני בתיקייה ובחר ב-Open in Terminal

טעינת התמונה במחשב אחר

docker load -i nsotask.tar

docker load טוען את התמונה השמורה בקובץ TAR לתוך המחשב כך שהיא תהיה זמינה למערכת של המשתמש.

התג -:i מציין את הקובץ שבו התמונה נשמרה, במקרה של הפרויקט הזה nsotask.tar.

כשמעבירים את קובץ ה-TAR למחשב אחר, צריך לטעון אותו לתוך Docker של המחשב החדש כדי שהוא יוכל להשתמש בתמונה ולהריץ אותה.

אם הפקודה לא תבוצע, התמונה לא תהיה זמינה במחשב החדש ולא תהיה אפשרות להריץ את הקוד בתוכה.

הרצת התמונה

docker run -it nsotask

docker run מריץ קונטיינר חדש מהתמונה שנבנתה במקרה של הפרויקט הזה התמונה היא nsotask.

התג -d: מציין שהקונטיינר ירוץ ברקע ולא יחסום את המסוף.

לאחר טעינת התמונה, אם המשתמש רוצה להריץ אותה (למשל, כדי להפעיל את הקוד בסביבה מבודדת), עליו להפעיל קונטיינר מהתמונה.

הפקודה docker run משמשת להריץ את התמונה כקונטיינר שמריץ את התוכנית שלך.

השימוש בתג -d עוזר להריץ את הקונטיינר ברקע כך שהמשתמש יכול להמשיך להשתמש במחשב מבלי שהקונטיינר יחסום את המסוף.

אם רוצים שהמידע שנשלח ישמר בקובץ אפשר להשתמש בפקודה הזאת:

docker run -it nsotask < name\_file.txt

הצגת הפלט של הקונטיינר

docker logs <container\_id>

docker logs מציגה את הפלט של קונטיינר שרץ. כאשר המשתמש מריץ את הקוד בתוך הקונטיינר, כל פלט שמופיע ב-console (כמו הודעות print) ייכתב בפלט של הקונטיינר.

<container\_id> הוא ה-ID של הקונטיינר שצריך להחליף בשם או ב-ID של הקונטיינר שבו המשתמש מעוניין לראות את הפלט.

דוגמא להרצת הפקודה: docker logs f3de26bac39e

לאחר שהקונטיינר רץ, יהיה אפשרות לראות את כל הפלט שהוא הדפיס (למשל, את תוצאות הביצוע של הקוד).

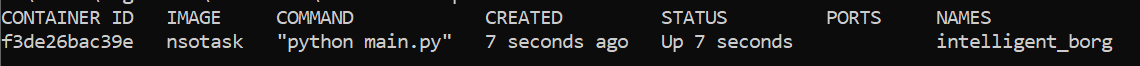
זה חשוב אם המשתמש רוצה לוודא שהקוד פועל בצורה תקינה, לבדוק אם יש טעויות, או לראות את הפלט שמתקבל בזמן הריצה של הקונטיינר.

מציאת ה-container id

בשביל למצוא את ה- container id אפשר להשתמש בפקודה docker ps.

הפקודה מאפשרת לראות דוקרים(dockers) שרצים במערכת.

לדוגמא:



כאן ה- container id הוא f3de26bac39e.