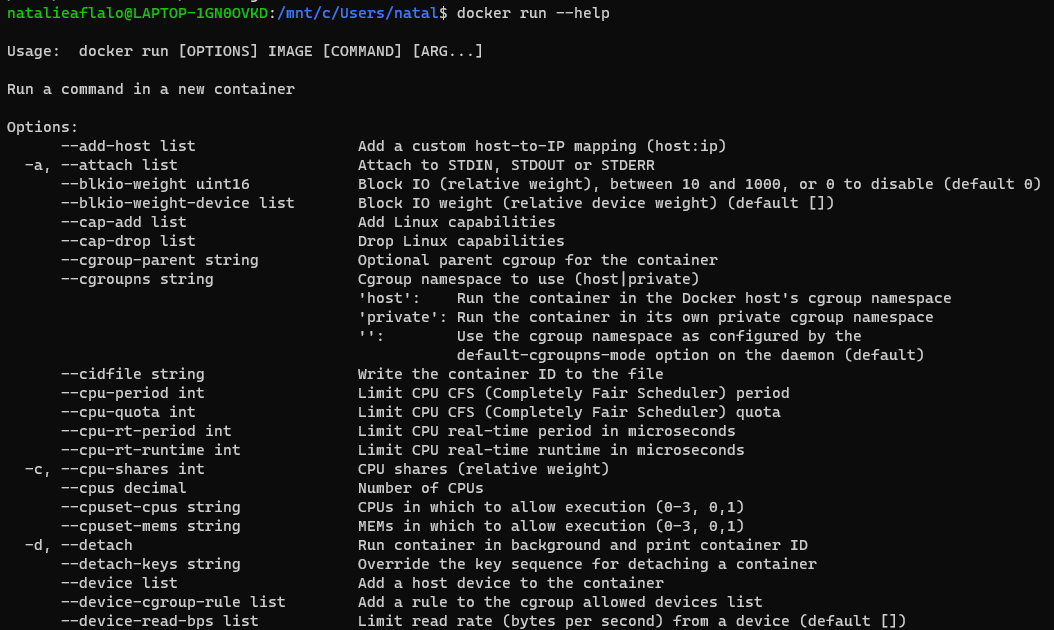
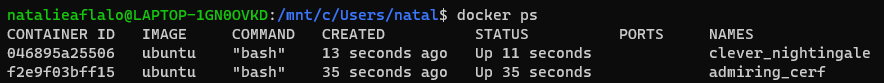
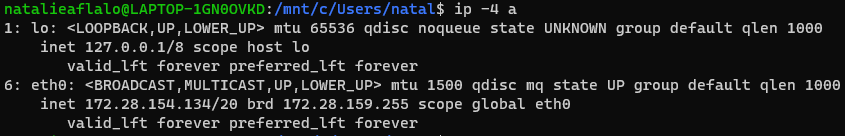
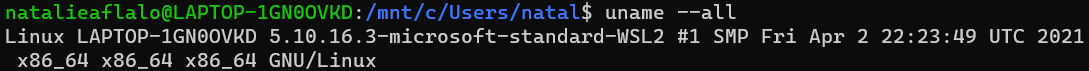
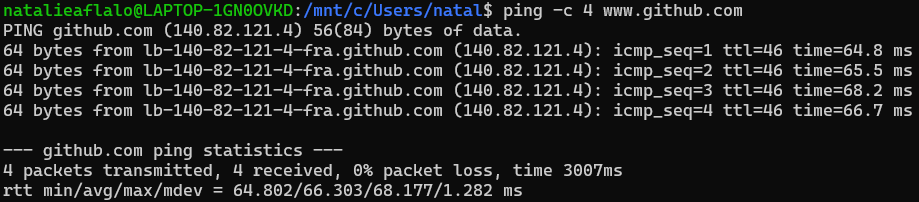
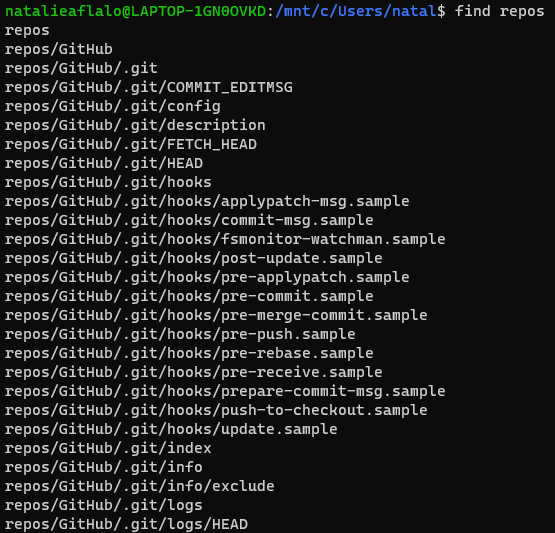
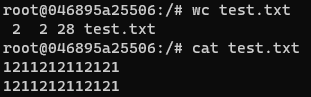
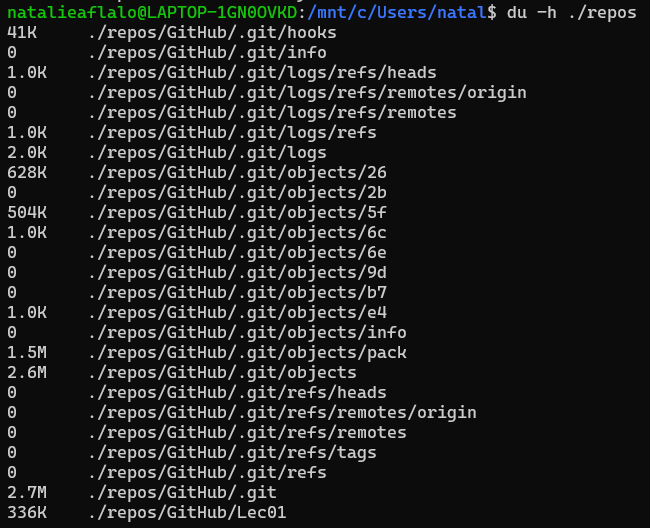
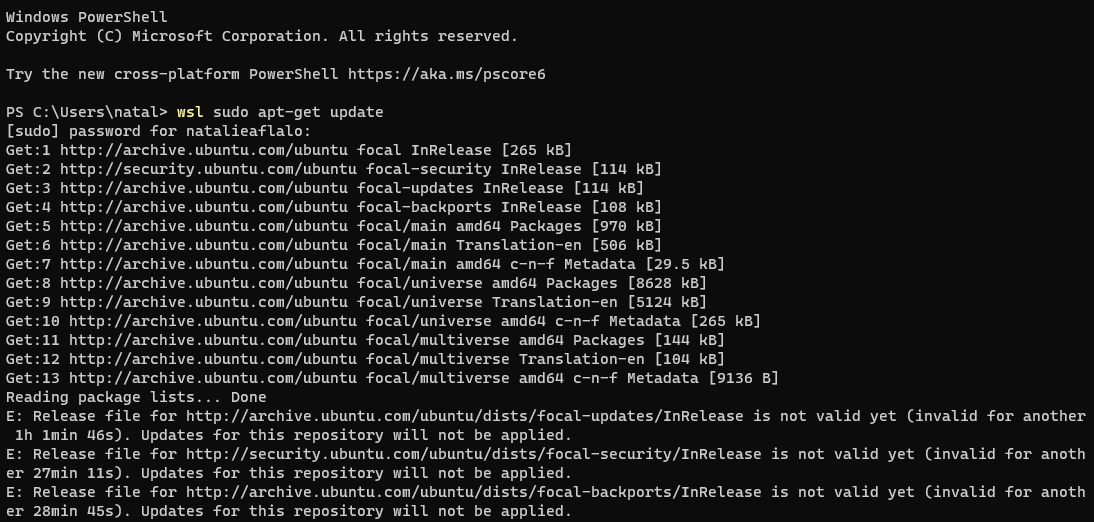
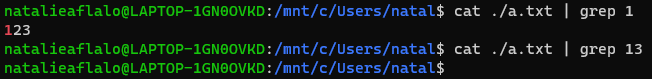
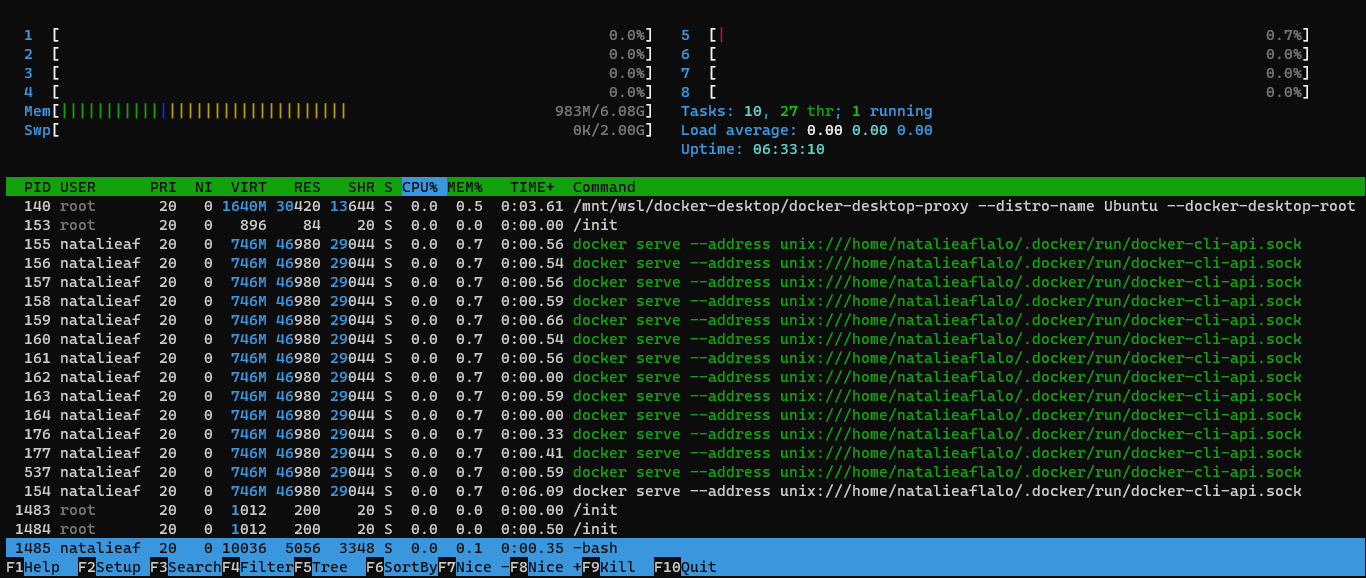
8.3.22

**הנדסת פתרונות תוכנה מתקדמים**

הבהרות והערות DOCKER ו-WSL:

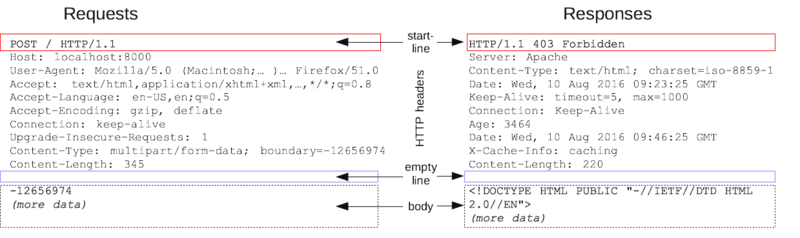
* Docker: שם של חברה שמספקת שירות שבו נלקח כלי (כמו מערכת הפעלה, סביבה של שפת תכנות וכו') בתצורת image ואותו "עוטפים" בcontainer שמריץ אותו.   
  האימג'ים יכולים להיות לוקליים (מותקנים על המחשב של המשתמש), ב-dockerhub או בשרת פרטי.
* Image- קובץ של הכלי (tool) כמו מערכת הפעלה.
* Container- התהליך של הרצת הimage.
* דוגמאות לפקודות ב-WSL – docker run (הרצת קונטיינר), echo (הדפסה).
* ניתן לתת לפקודות פרמטרים (options) כך:
* ניתן להוסיף קיצור לoption בעזרת dash (מקף -), כלומר לאחר dash תופיעה אות אחת שמייצגת פרמטר מסוים. לדוגמה -v הוא קיצור של --volume. ניתן לרשום גם בצורה הקצרה וגם בצורה הארוכה. את הקיצורים ניתן לשרשר תחת אותו dash כך: -tidp כלומר יש כאן 4 options שונות תחת אותו dash. (option שצריכה לקבל ערך, כמו p שמקבלת פורטים, עדיף לא לשים בשרשור כזה. אם שמים היא צריכה להיות אחרונה)
* עבור options שאין להם קיצור או שנרצה לפנות אליהם בשם המלא נצטרך לתת double dash (שני מקפים אחד אחרי השני --). לדוגמה --help, --ip.

כדי לדעת איזה options קיימות לפקודה מסוימת נרשום אותה ואחריה --help ותופיעה רשימה. במידה ויש קיצור לפקודה הוא יופיע משמאל לפקודה.

* משפטים:
* Image אחד יכול לרוץ במקביל בכמה containers.
* ב-container אחד יכול לרוץ רק image אחד.
* פקודות נוספות ב-WSL:
* ip : מאפשרת להציג ולשנות נקודות גישה לרשת (network interface)
* ifconfig : מאפשרת להציג ולשנות נתונים של network interface במערכת (כמו כתובת IP)
* hostname : מציג את שם המחשב הלוקלי  
  
* uname : מחזירה מידע בסיסי על מערכת ההפעלה (linux)  
  
* ping : מאבחן ובודק תקשורת לרשת  
  
* find : משמש למציאת תיקיות וקבצים  
  
* wc : פקודה שסופרת שורות וכמות תווים.  
  
* xxd : מציג נתונים מקובץ בינארי בבסיס הקסדצימלי. כדי לצאת מהמצב הזה יש ללחוץ q
* du -h / : מציגה ניצול הזיכרון בדיסק של תיקייה או קובץ   
  
* chown : שינוי של משתמש "בעלים" של תיקייה, קובץ או קישור
* chmod : מאפשר לשלוט במי יכול לגשת לקובץ או לחפש בתיקייה
* sudo : פקודה שמריצה תהליך נוסף במקביל עם משתמש אחר (לרוב משתמש admin) ולכן גם תדרוש סיסמה.
* which : מוצאת נתיב של פקודה  
  
* grep : מוצא את כל המקומות בנתיב שיש בהם את הערך שניתן לו. אם לא קיים הערך הוא לא מחזיר כלום. אם נוסיף -v יציג את כל השורות ללא הערך שניתן לו.  
  
* jobs : מציג תהליכים שרצים ברקע  
  
* fg : נכנס חזרה למסך של הjob. ניתן לצאת בעזרת ctrl+z
* bg : ה-job ממשיך לרוץ ברקע.
* kill : פקודת תקשורת (שליחת סיגנלים) בין תהליכים. בין היתר הורג תהליכים.  
  
* htop : מאפשר בקרה בזמן אמת על תהליכים ומשאבים של המערכת.
* top : כמו htop אך פחות נוח לקריאה וללא אפשרות גלילה עם העכבר.

**תקשורת בין מיקרוסרוויסים:**

* API- ממשק תכנות יישומים. ערכת ספריות קוד, פונקציות, פקודות וכו' של יישום, שניתן לעשות בה שימוש פשוט כדי לנצל את התוכן שלה ליישום שלנו. (לדוגמה, אם נשתמש ב-API של יישום אחר, נוכל להשתמש בפונקציות שלו מבלי לכתוב אותן ביישום שלנו)
* HTTP API- ממשק תכנות יישומים של שרת רשת או דפדפן, מאפשר שימוש בAPI ללא חשיפת הקוד של היישום. התקשורת מבוססת על בקשה request ותגובה response.
* REST API- כמו HTTP , בנוסף מציג את הstate שלו בשקיפות. <https://www.youtube.com/watch?v=lsMQRaeKNDk>
* קובץ json- קובץ בו הנתונים מסודרים בתצורת dictionary , כלומר מידע המאוחסן בתצורה של מפתח-ערך.

תקשורת של מיקרוסרוויסים בינהם היא over HTTP, כלומר באמצעות API, מעבירים בינהם json-ים.

HTTP REQUEST:

|  |  |
| --- | --- |
| **The HTTP method** | **The operation we want** |
| POST | Create |
| GET | Read |
| PUT | Update |
| DELETE | Delete |

Endpoint: מי שאנחנו פונים אליו בrequest. לדוגמה, <https://www.google.com>. נציין כי אותו endpoint יכול לקבל בקשות GET וגם בקשות POST.

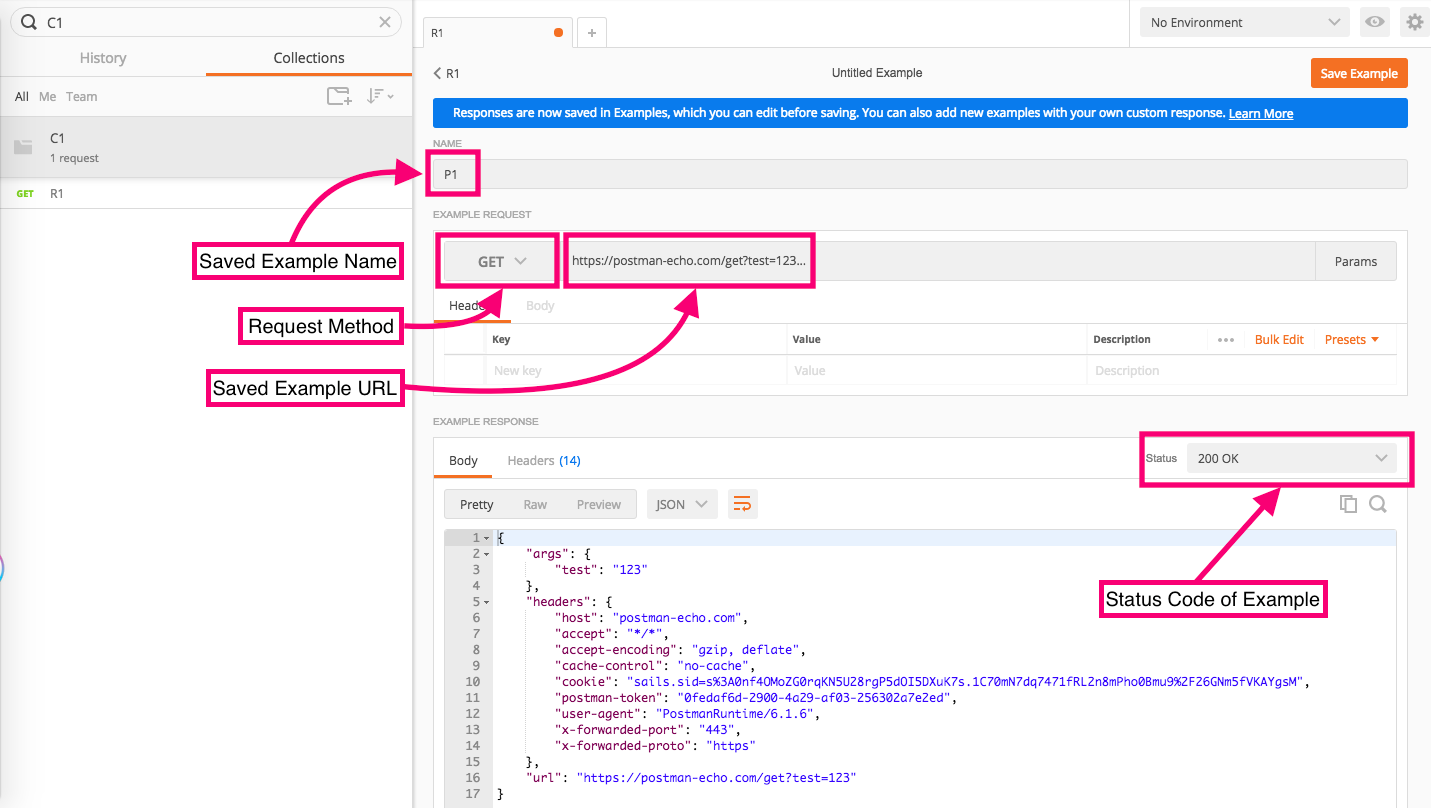
* GET: מבקש להציג מידע מהמקור שאליו אנחנו פונים.  
  GET /microservice/v1/function?param1=value1&param2=value2  
  סימן השאלה הוא הsplitter בין המקור אליו אנו פונים לבין הפרמטרים שלפיהם נרצה לקבל את המידע. כתובת המקור (משמאל לסימן השאלה) היא חד-חד-ערכית.
* POST: מעבירה מידע, קובץ, מחרוזת וכו' שרוצים לעדכן במקור אליו פונים. בבקשה זו ניתן לתת כמות גדולה יותר של פרמטרים ואת כל המידע הזה מעבירים בbody של הבקשה. (דוגמה בתמונה למעלה)

פלטפורמות ליצירת בקשות HTTP: postman, פקודות curl ו-wget ב-command line (מוסבר בסיכום של הרצאה 3).

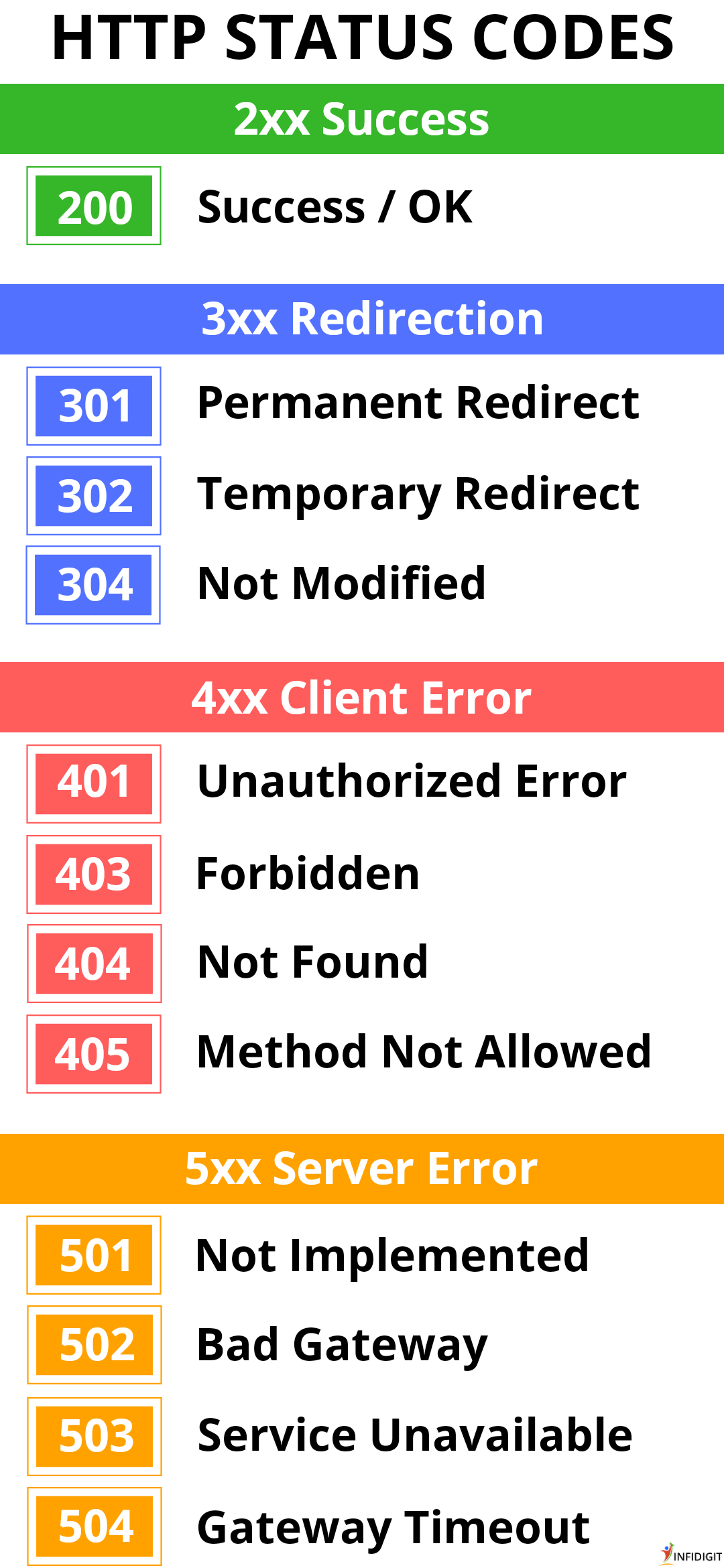
דוגמה לבקשה בעזרת curl: (ניתן להחליף את GET ב-POST)



דוגמה לבקשה בpostman:



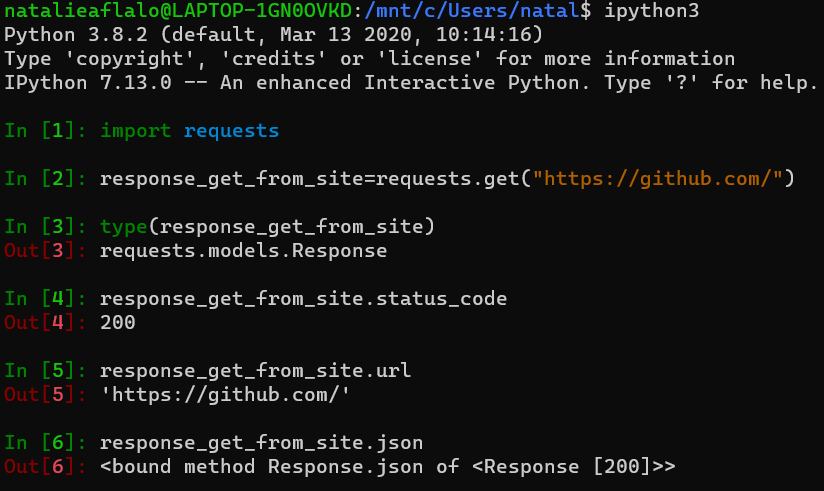
HTTP RESPONSE:



בקשות HTTP בפייתון-

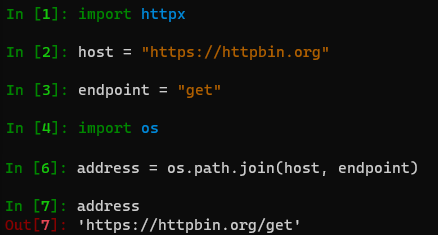
ספריית requests:

נבצע import requests (או pip install requests אם לא מותקנת)

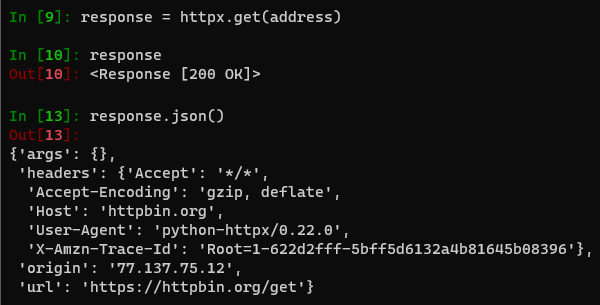


ספריית httpx:

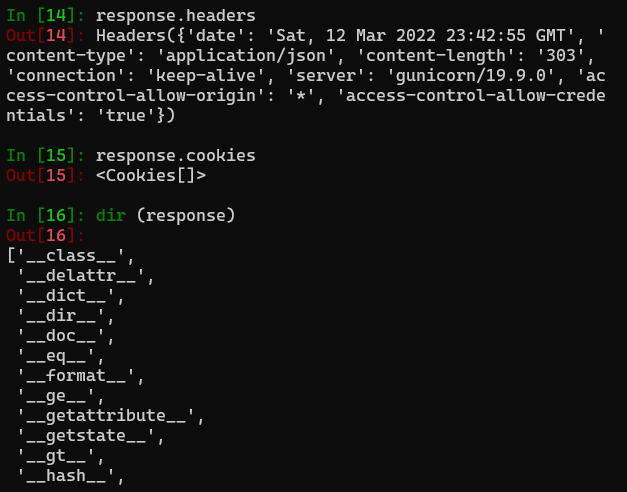
נבצע import httpx (או pip install httpx אם לא מותקנת)



כעת נבצע את פקודת ה-GET ונצפה בתוצאה בתצורת json:



פונקציות נוספות שניתן להריץ על ה-response:



Cookies- קבצים שנשמרים בדפדפן של הלקוח ומאפשרים התחברות מהירה, שמירת העדפות וכו'.