**Final Task**

מגישים:

רביד שלו - 207479379

טלפון - 0508788042

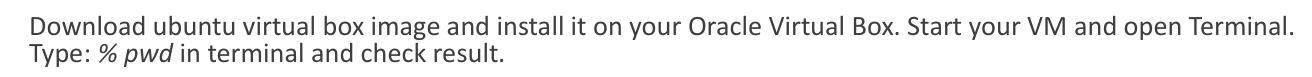
אימייל: ravidxcx@gmail.com

עמית דרעי - 322467283

טלפון - 0525110204

אימייל: amitderei123@gmail.com

הרצאה 1. intorductio

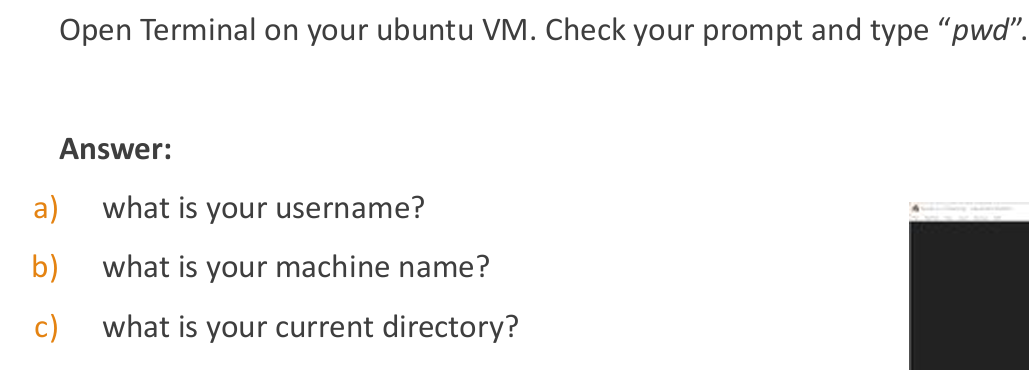


תמונה שמכילה טקסט, צילום מסך

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

הרצאה 2. linux

שקופית 10:



Username: ubadmin

Machine name: Ubuntu22

Current directory: /home/ubadmin

תמונה שמכילה טקסט, צילום מסך

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

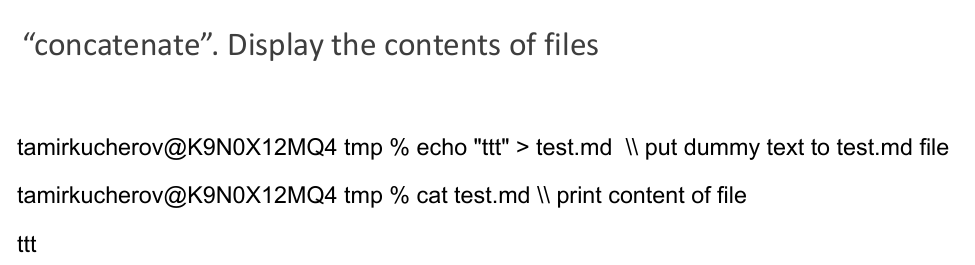
שקופית 26:

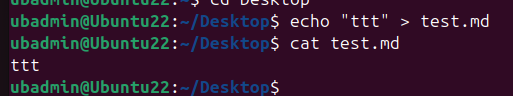


תמונה שמכילה טקסט, צילום מסך

תוכן בינה מלאכותית גנרטיבית עשוי להיות שגוי.

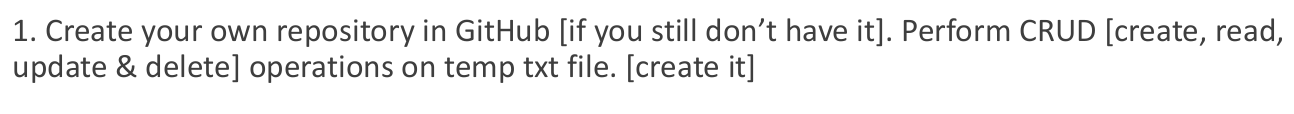
שקופית 35:



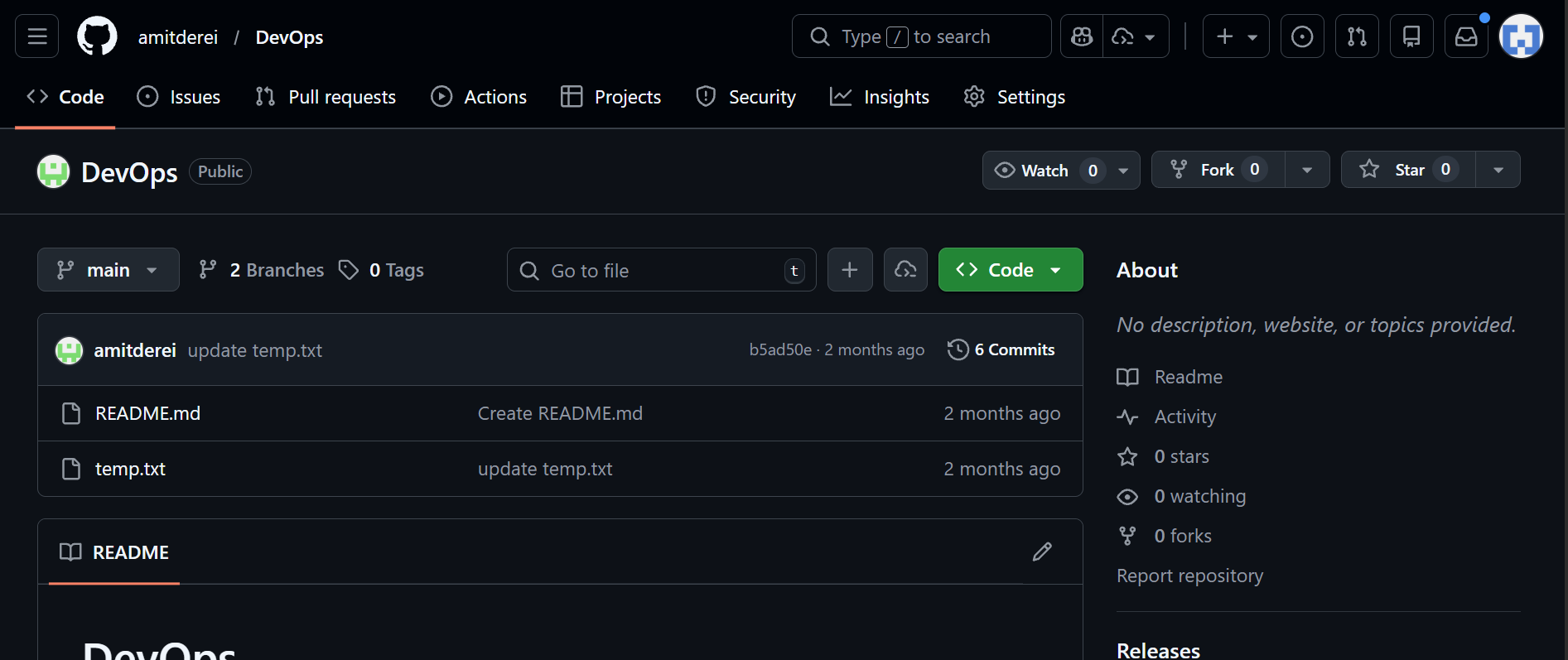


הרצאה 3. Git:

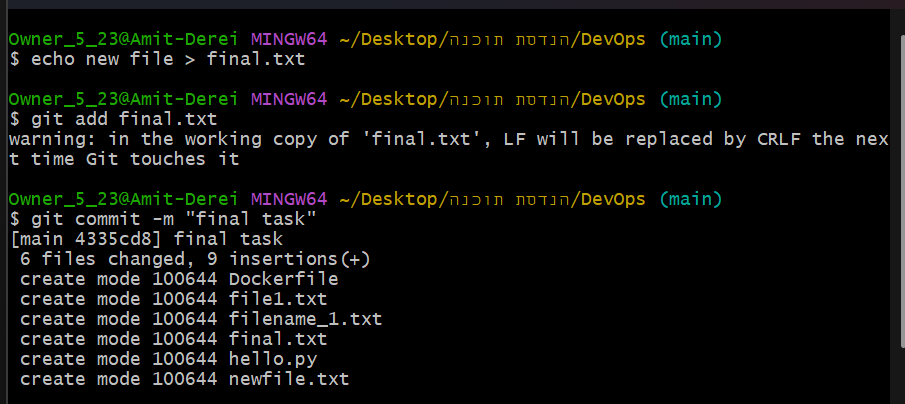
שקופית 46:



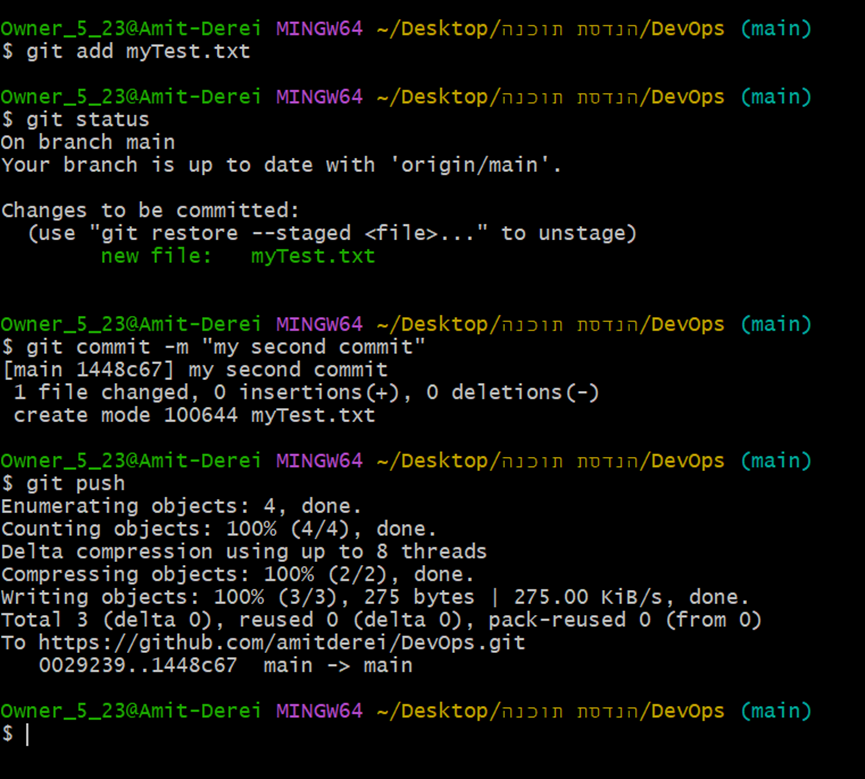
Create our own repo.



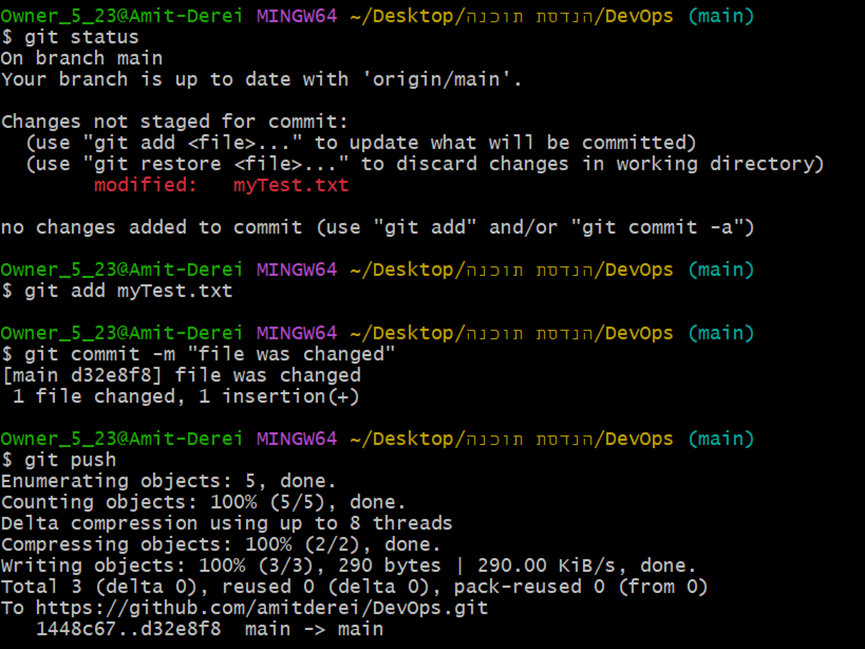
Create file:



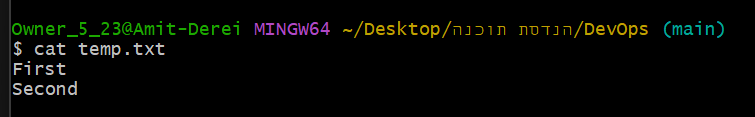
Add new file



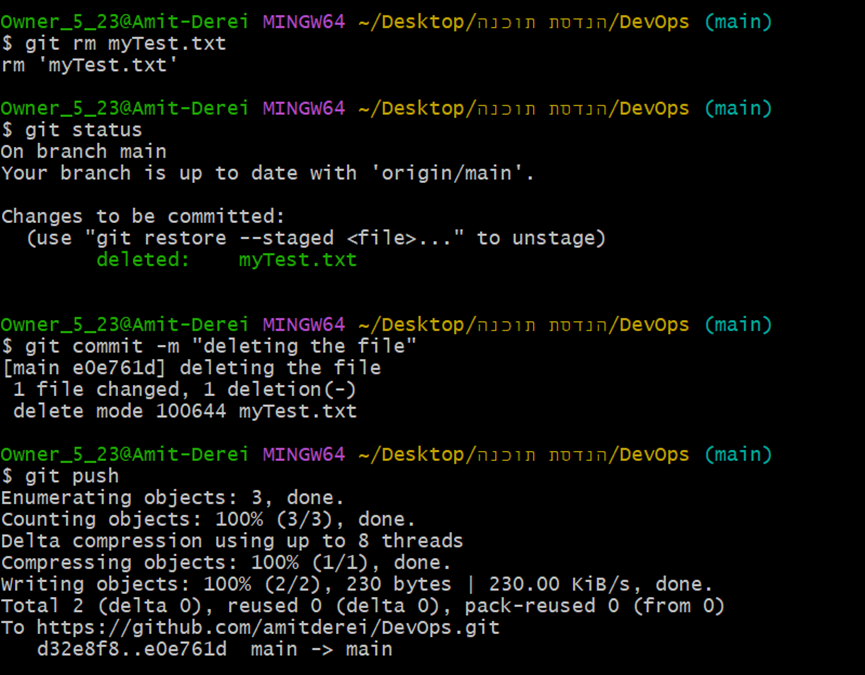
After change the File:



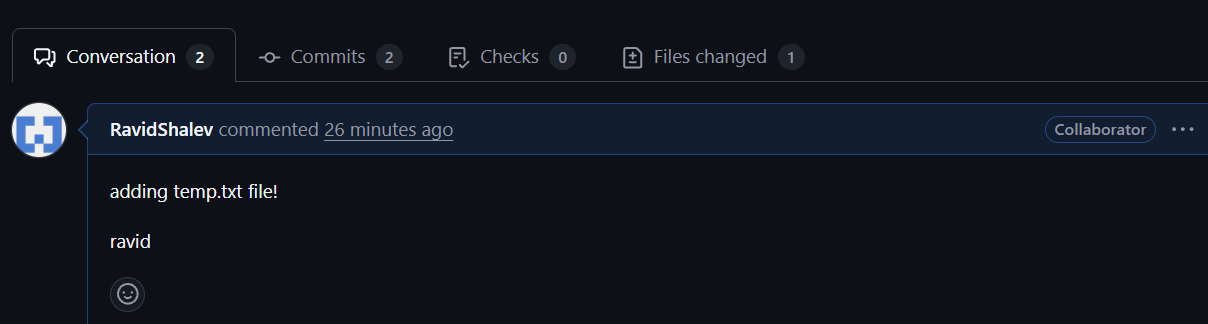
read: with cat

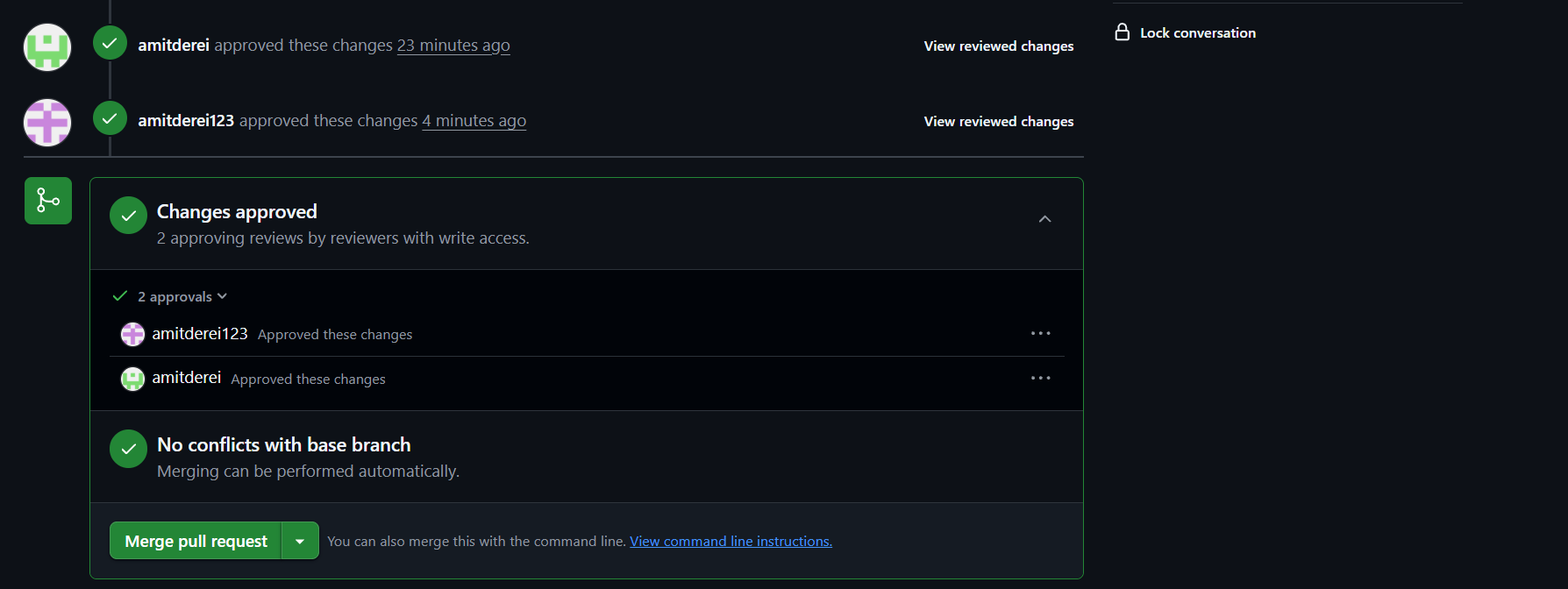


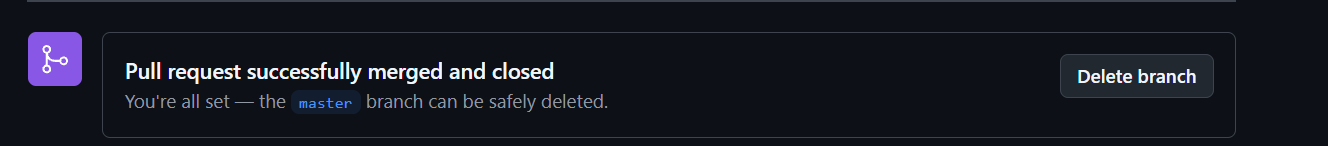
delete: with rm



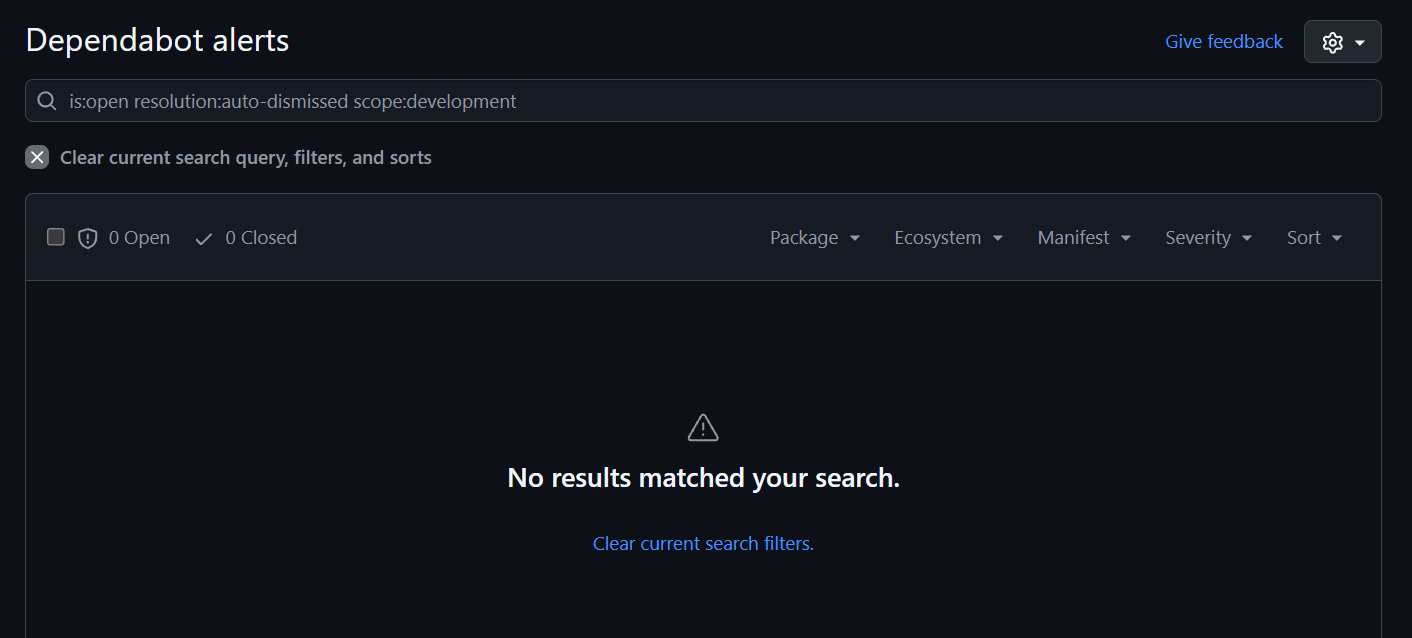
PullRequest with 2 approvers:







**לגבי Dependabot:**



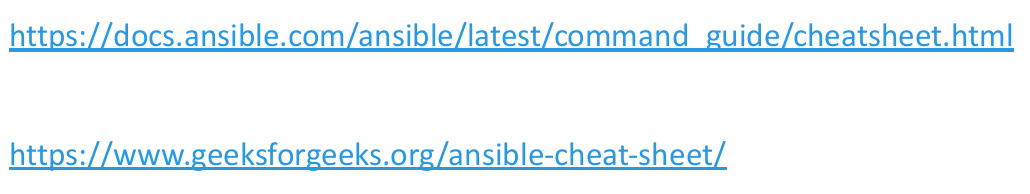
**1. למה צריך?** כדי לבצע אוטומציה לעדכוני אבטחה. במקום שנבדוק ידנית כל הזמן אם יצא עדכון לספריות שלנו, הוא עושה את זה בשבילנו.

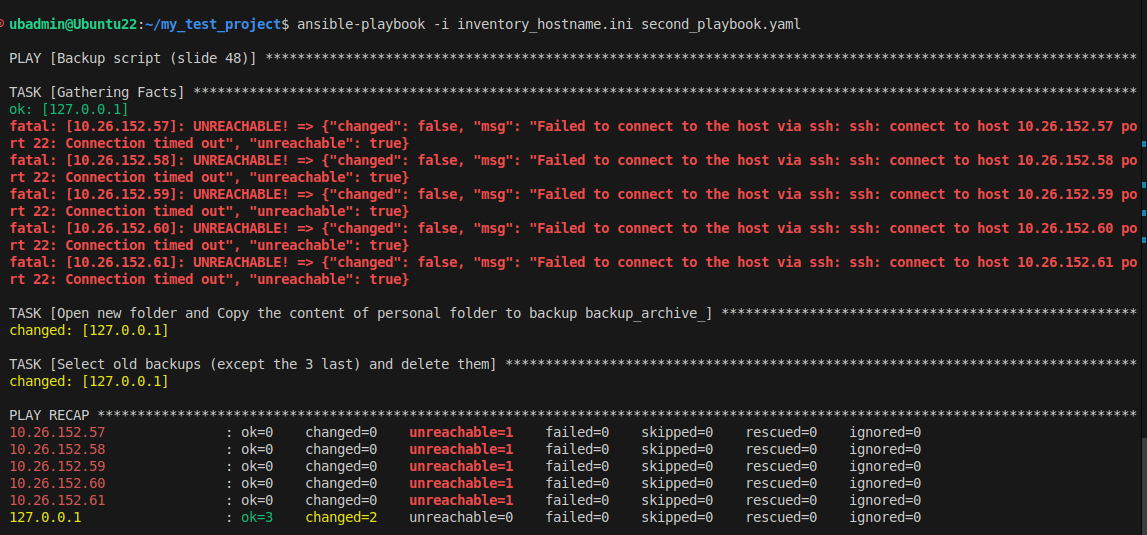
**2. מתי משתמשים?** אנחנו לא מפעילים אותו, הוא עובד ברקע באופן קבוע. הוא נכנס לפעולה אוטומטית ברגע שמתגלה חולשת אבטחה (Vulnerability) או יוצאת גרסה חדשה לספרייה בפרויקט.

**3. למה זה חשוב?** בעיקר בגלל אבטחת מידע (Security). רוב הפריצות קורות דרך ספריות לא מעודכנות. וה-Dependabot סוגר את הפרצות האלו לפני שהאקרים מספיקים לנצל אותן.

הרצאה 4. Ansible

שקף 47: הרצת הקוד

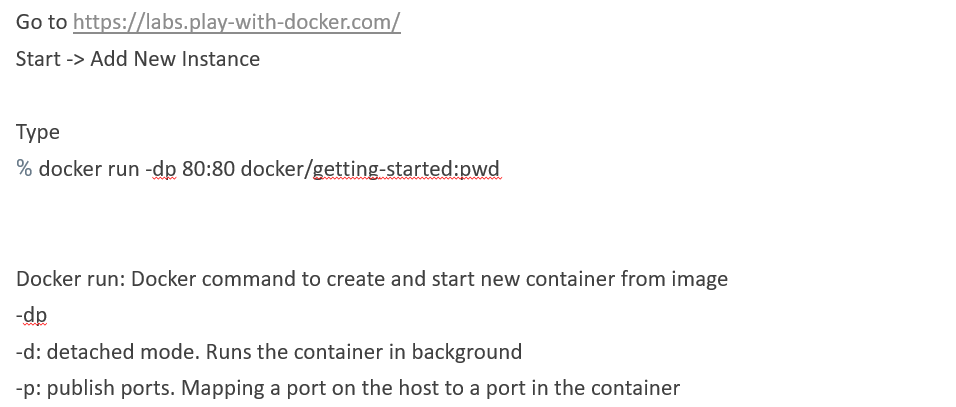


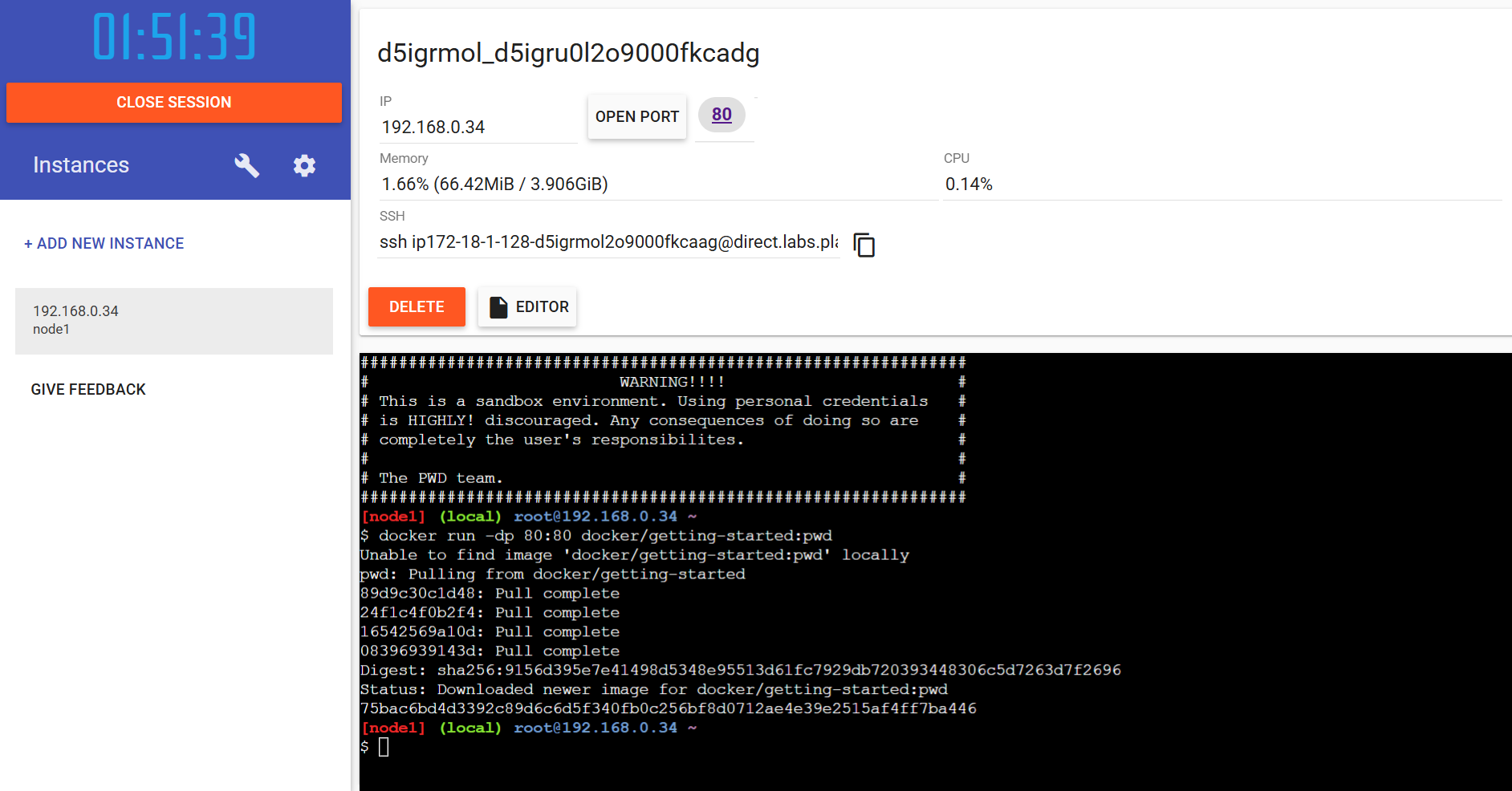


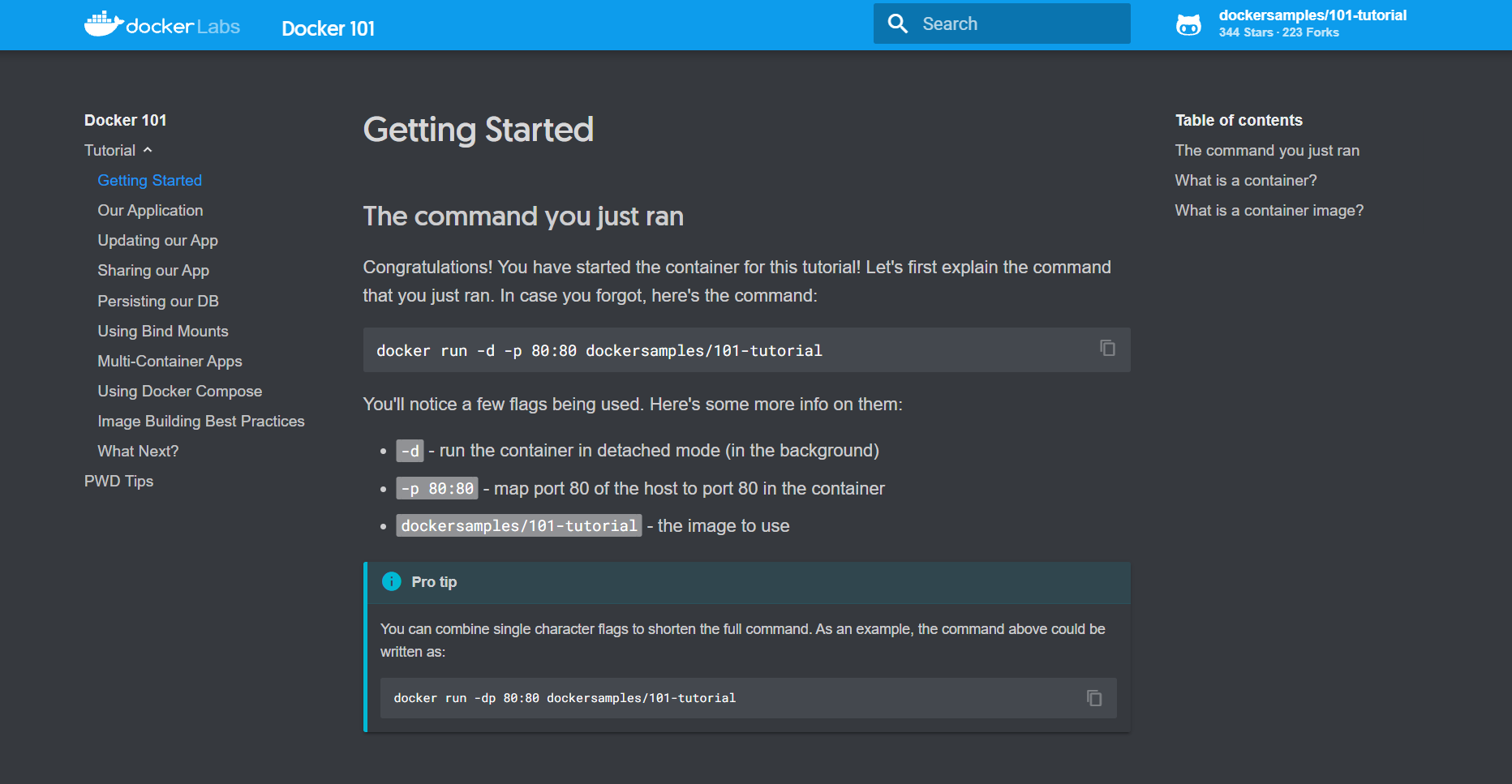
ניתן לראות שהצליח להריץ על המחשב הלוקאלי, על המחשבים המרוחקים הוא לא הצליח משום שאין לנו גישה אליהם.

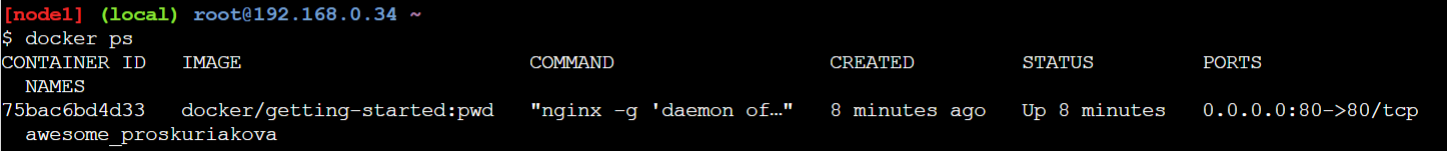
הרצאה 5. docker:

שקופית 29:





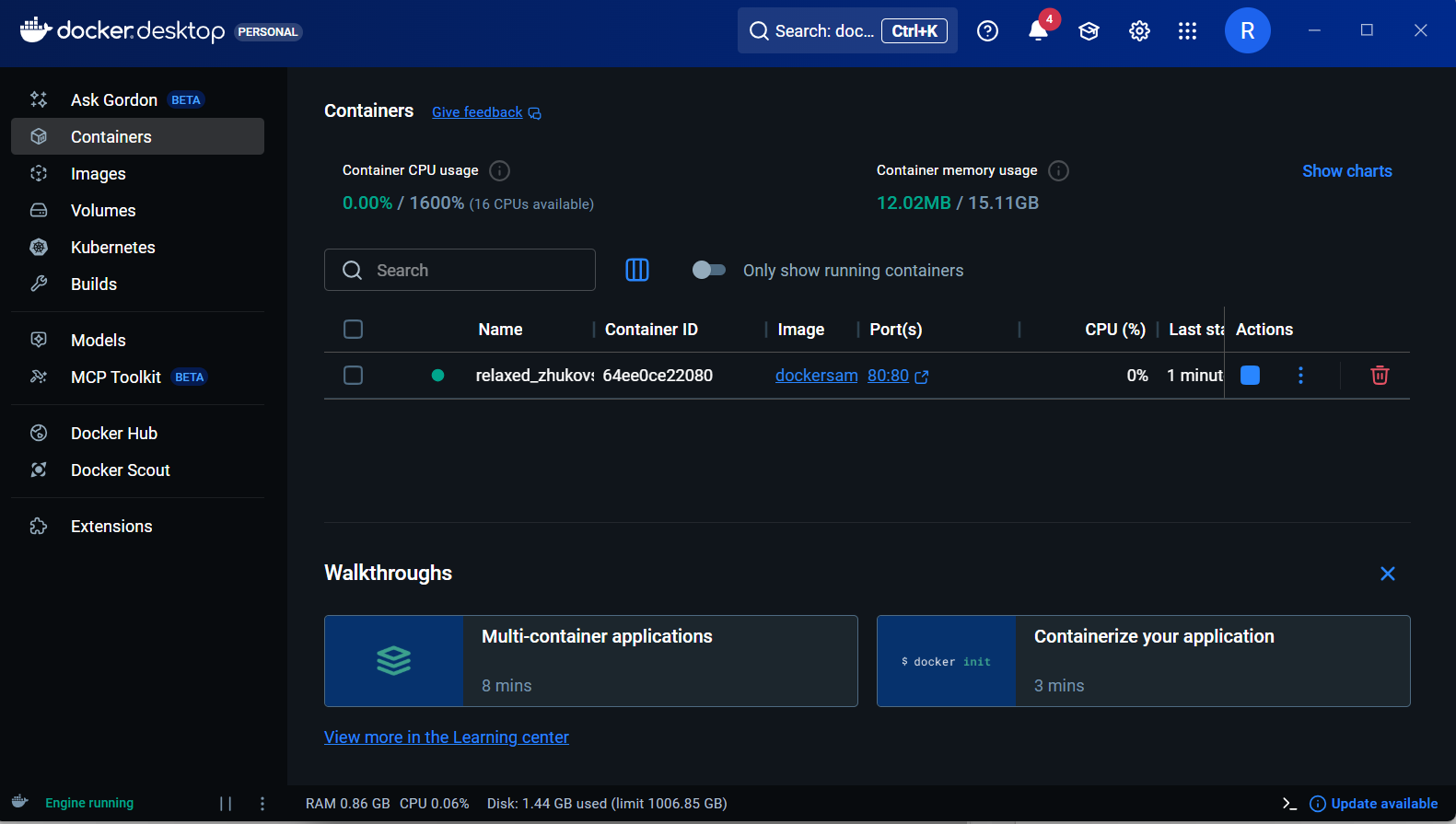


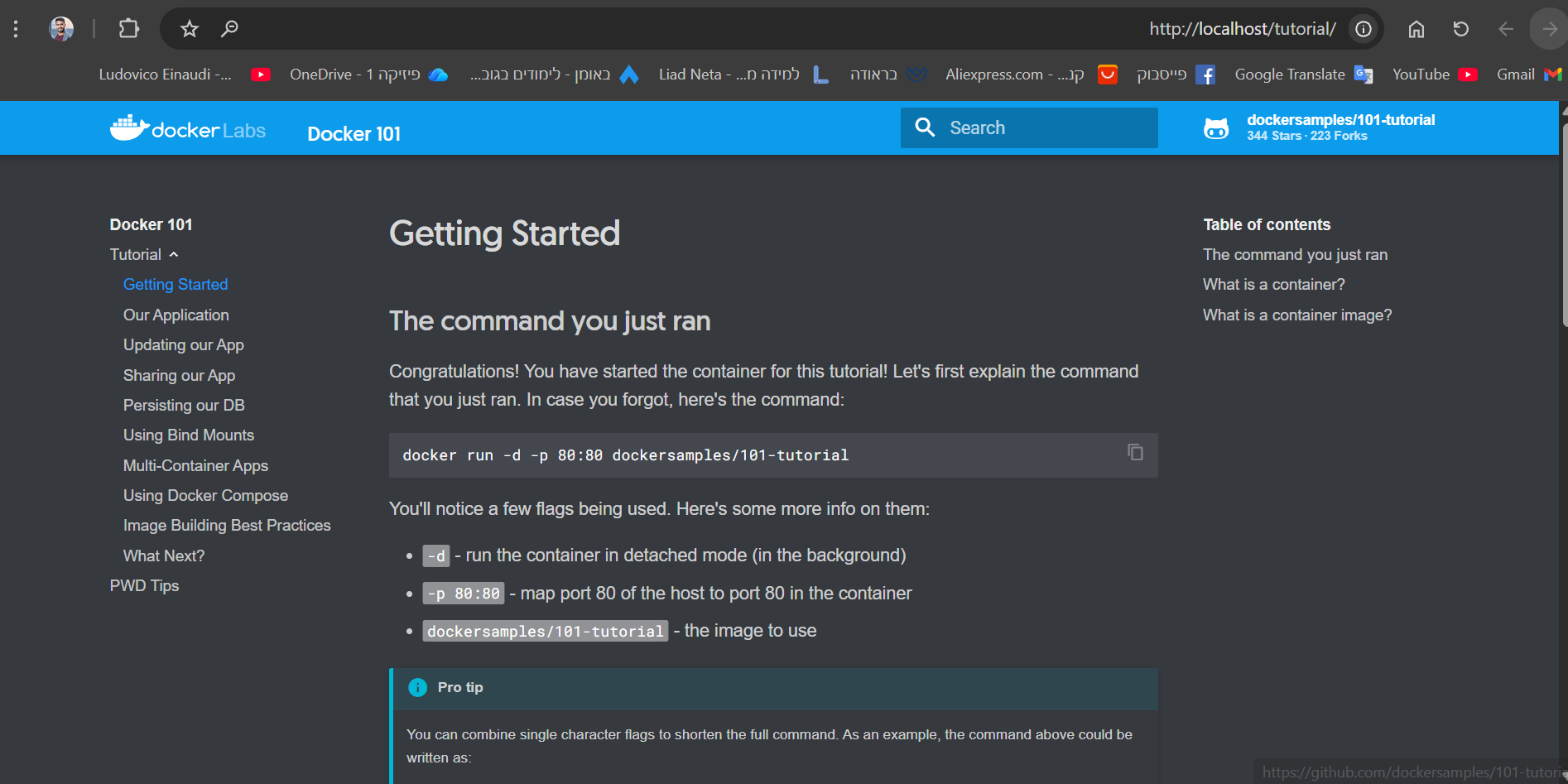


שקופית 33:

הרצה מ docker desktop

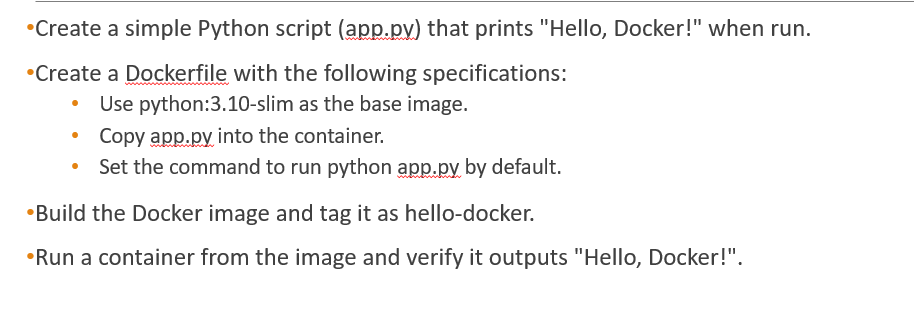


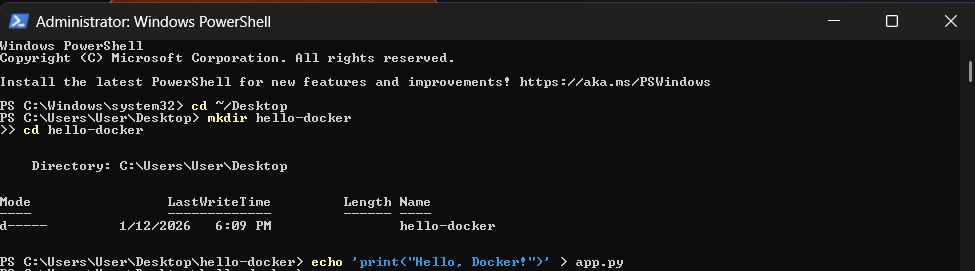




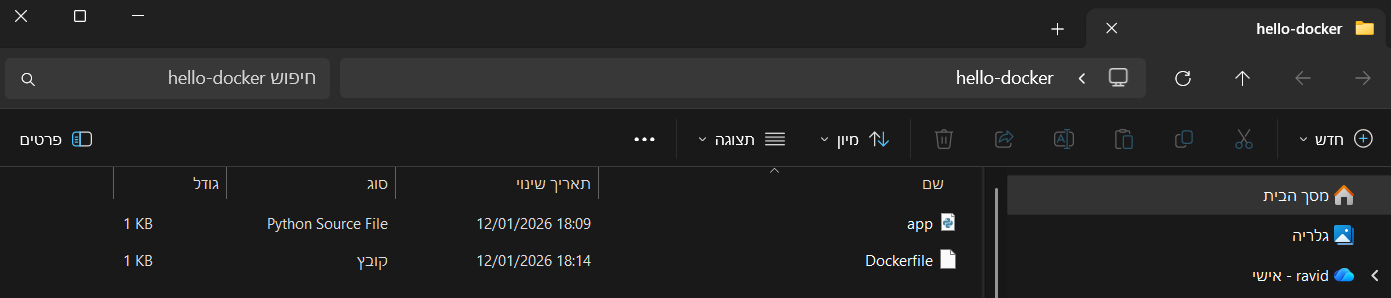
ניתן לראות שהתמונה רצה בdesktop ואם לוחצים על ה port 80:80 נפתח לי הגוגל כרום עם localhost כמו שמוצג להלן.

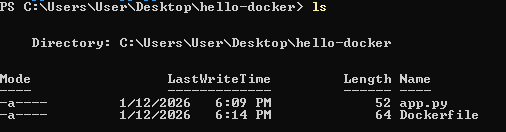
שקופית 45:

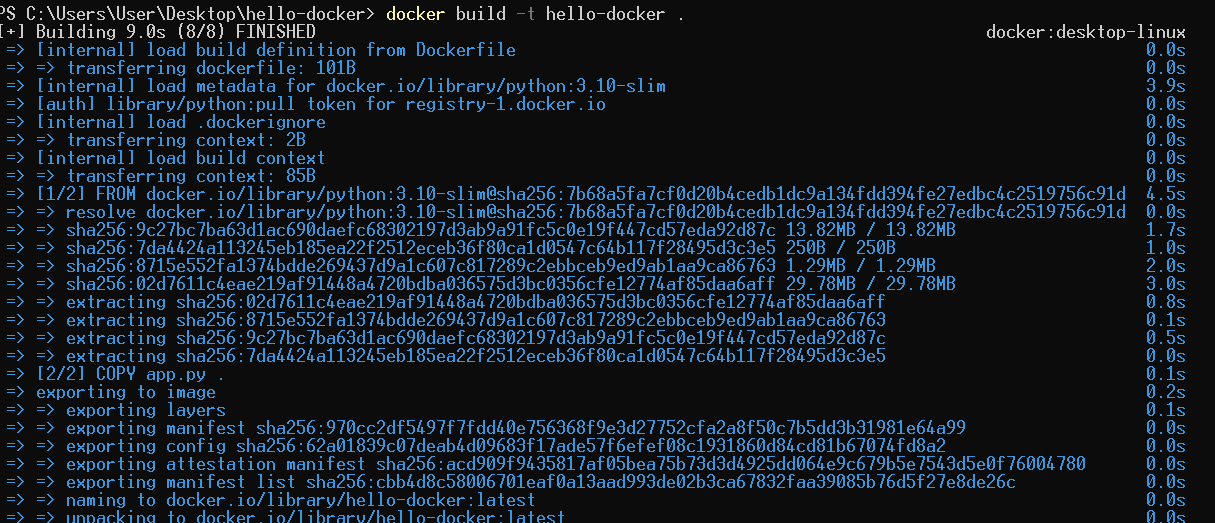




****

****

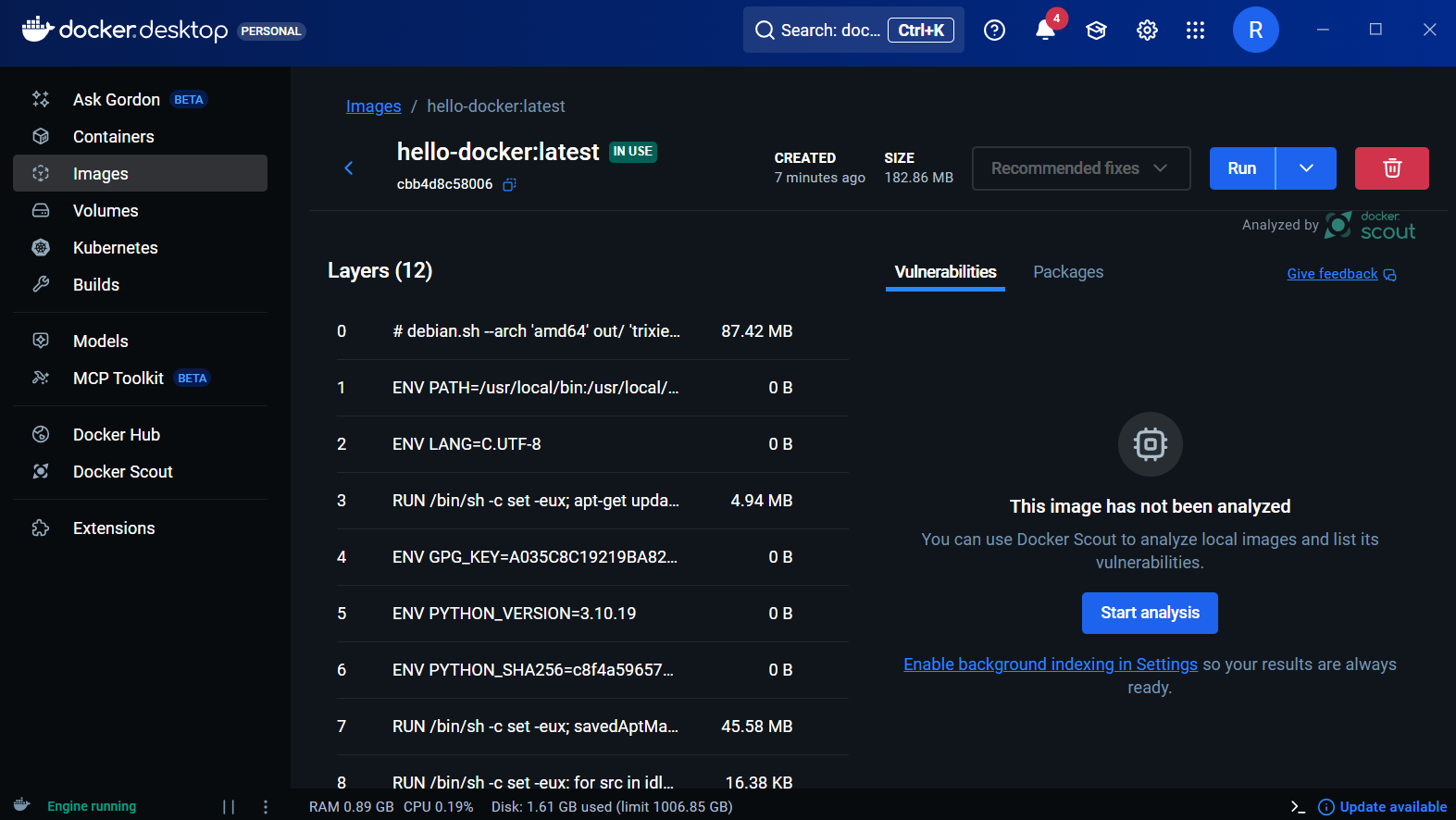
****

עכשיו אחרי שיצרנו את הקבצים של פייתון ודוקר ניצור את התמונה:  


הנקודה . בסוף היא חלק מהפקודה היא אומרת לדוקר שהקבצים נמצאים כאן בתיקייה לפי מה שהבנו בכיתה.

כעת נריץ את הקונטיינר אחרי שהbuild יסתיים



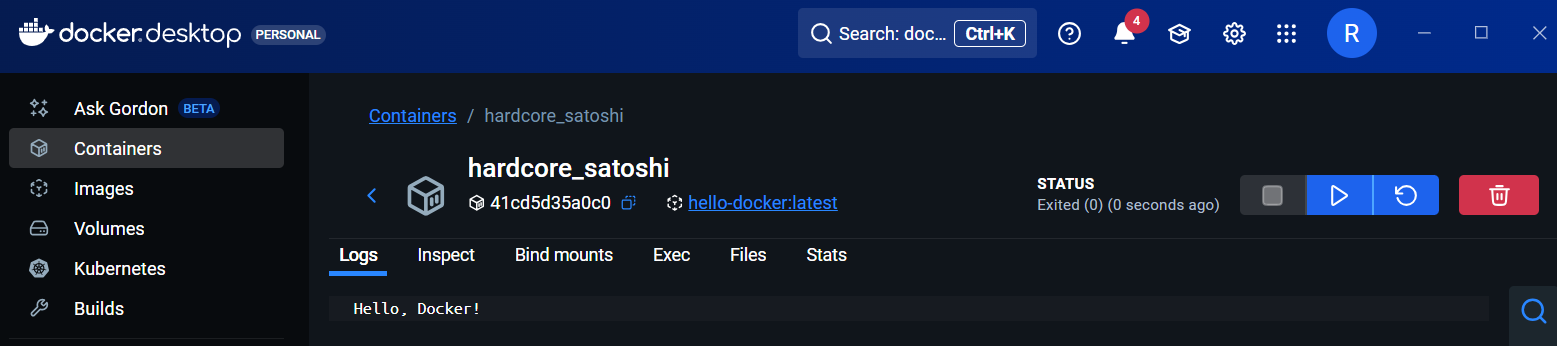
אפשר כמובן גם דרך הGUI:  




נעשה גם מ-docker desktop







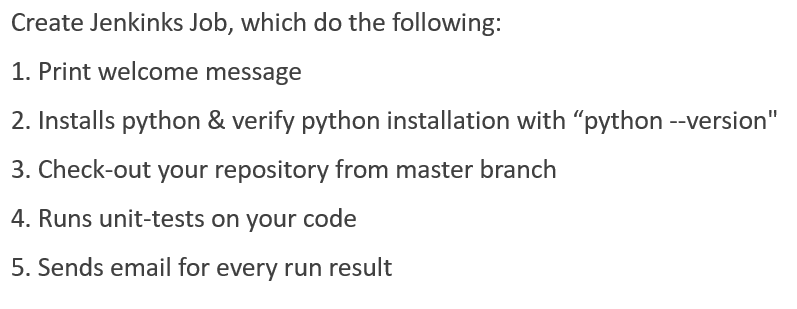
הרצאה 6. jenkins:

**הכנה:** מכיוון שההרצה מתבצעת דרך Docker, יש לוודא שה־Docker image שבו Jenkins משתמש כולל התקנת Python וכל התלויות הנדרשות.





שקופית 19:

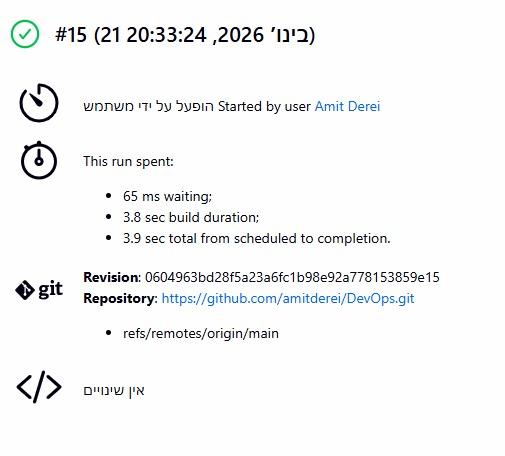


מוסיפים את הריפו לעבודה

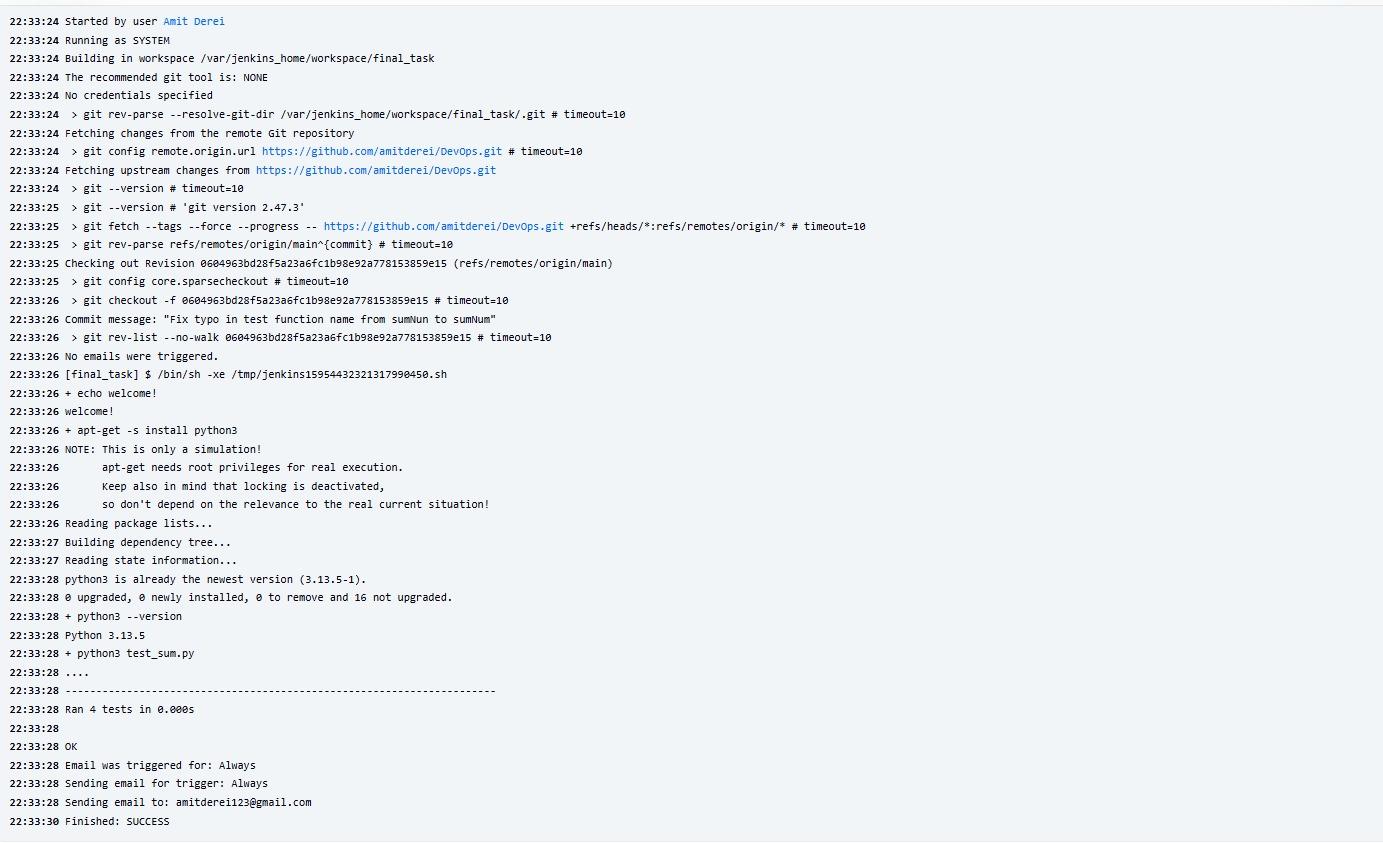
מכילה 2 קבצים, שאותם נצרף לקובץ ([sumNumbers.py](http://sumnumbers.py), test\_sum.py)

[sumNumbers](http://sumnumbers.py)- קיימת בתוכה פונקציה שמקבלת מספרים a, b ומחזירה חיבור ביניהם.

test\_sum.py- מכיל בדיקות יחידה לפונקציה המבצעת חיבור בין 2 פרמטרים.



הלוג שהודפס (console ouput)



הרצת shell לעבודה הזאת



#print a welcome message to the console (1)

echo "welcome!"

# Python is already installed during the initial setup because Jenkins is running inside Docker,

# so this step only checks/simulates the installation (2)

apt-get -s install python3

#check that Python is installed and print its version

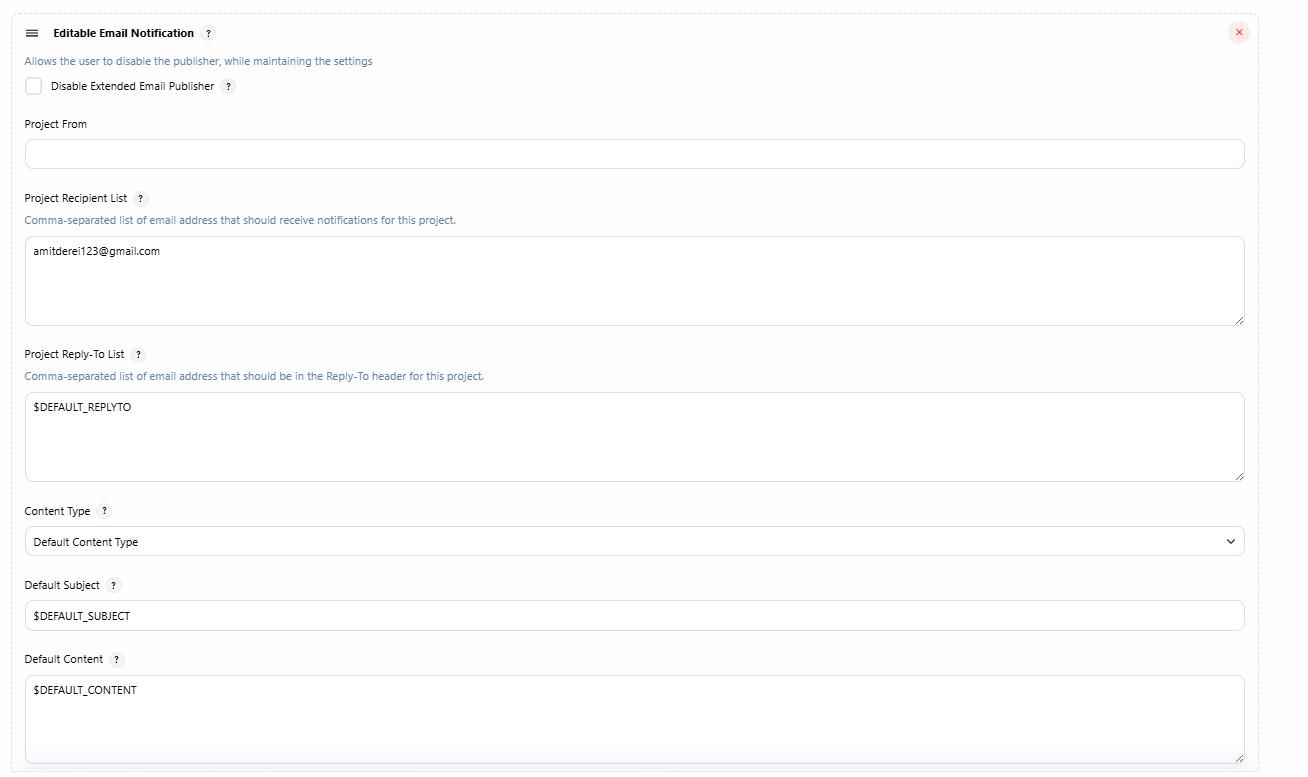
python3 --version

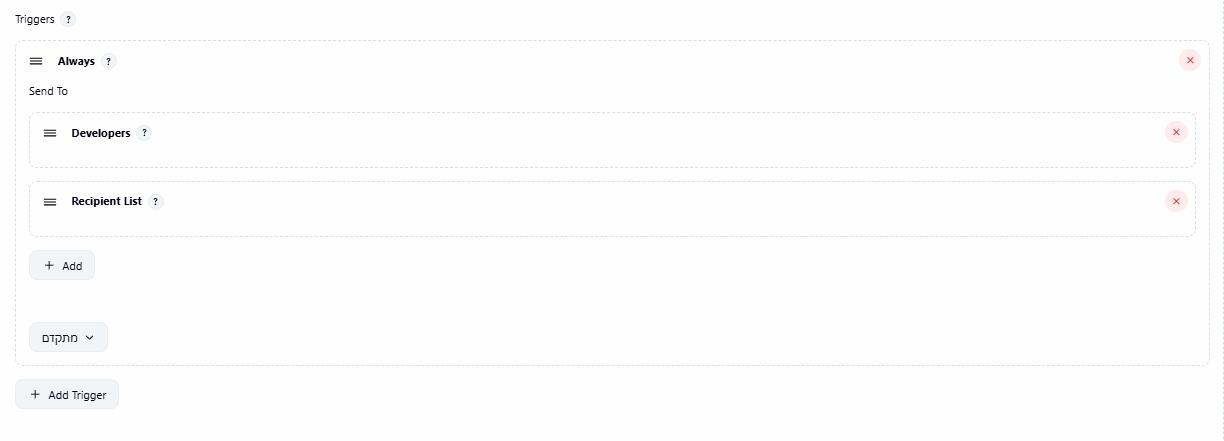
# Jenkins performs git clone automatically (3)

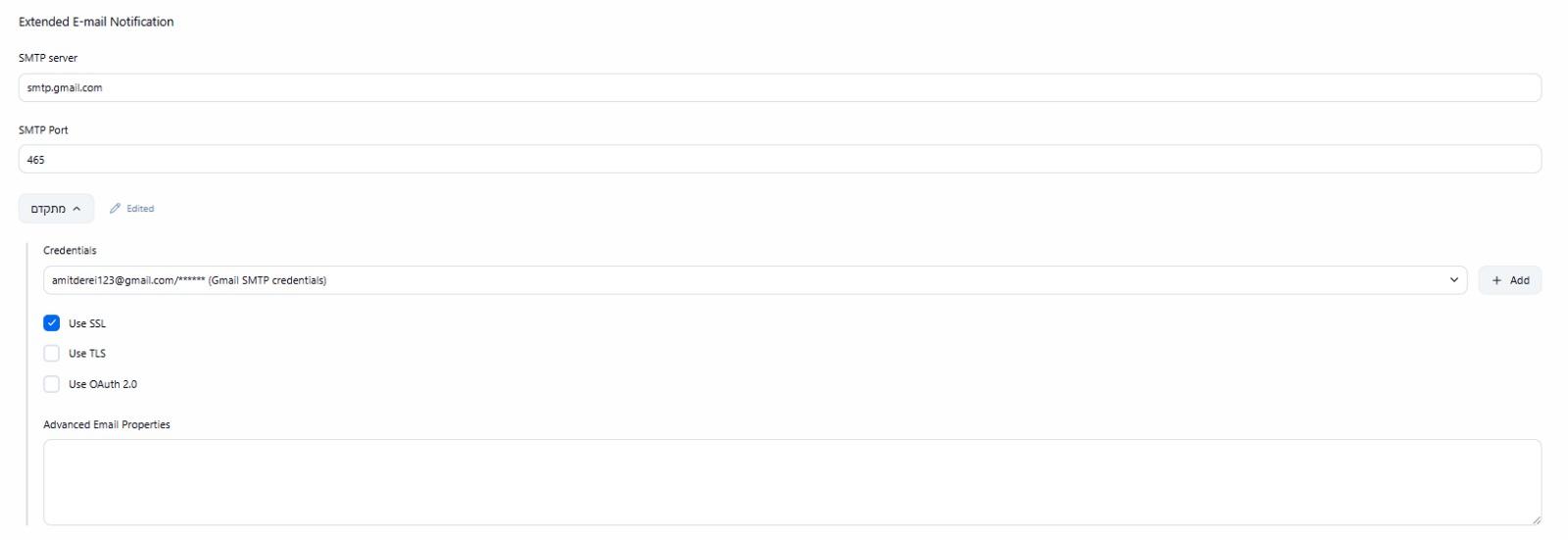
#run unit test

python3 test\_sum.py

יצירת אימייל notification - מגדירים גם בהגדרות של הjenkins את השרת שמשתמשים בו ואת המפתח שקיבלנו מהגימייל. בנוסף, נגדיר טריגר לשאחר כל הרצה בין אם היא הצליחה ובין אם לא נקבל מייל.



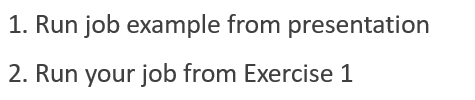


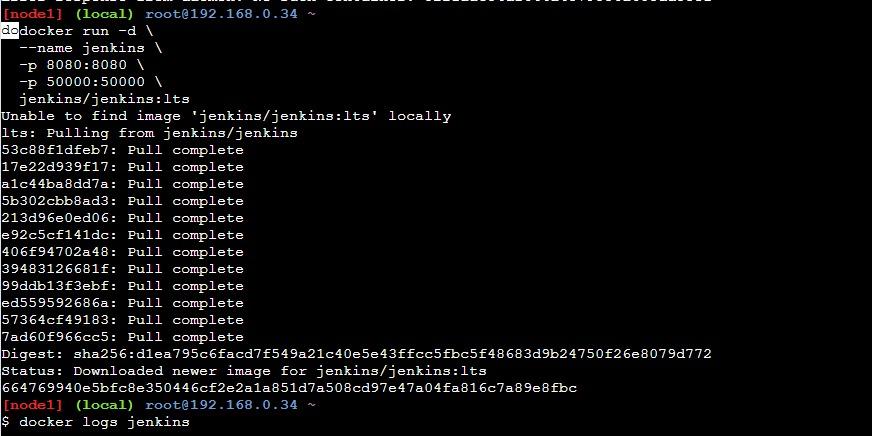


המייל שהתקבל שהצליח!

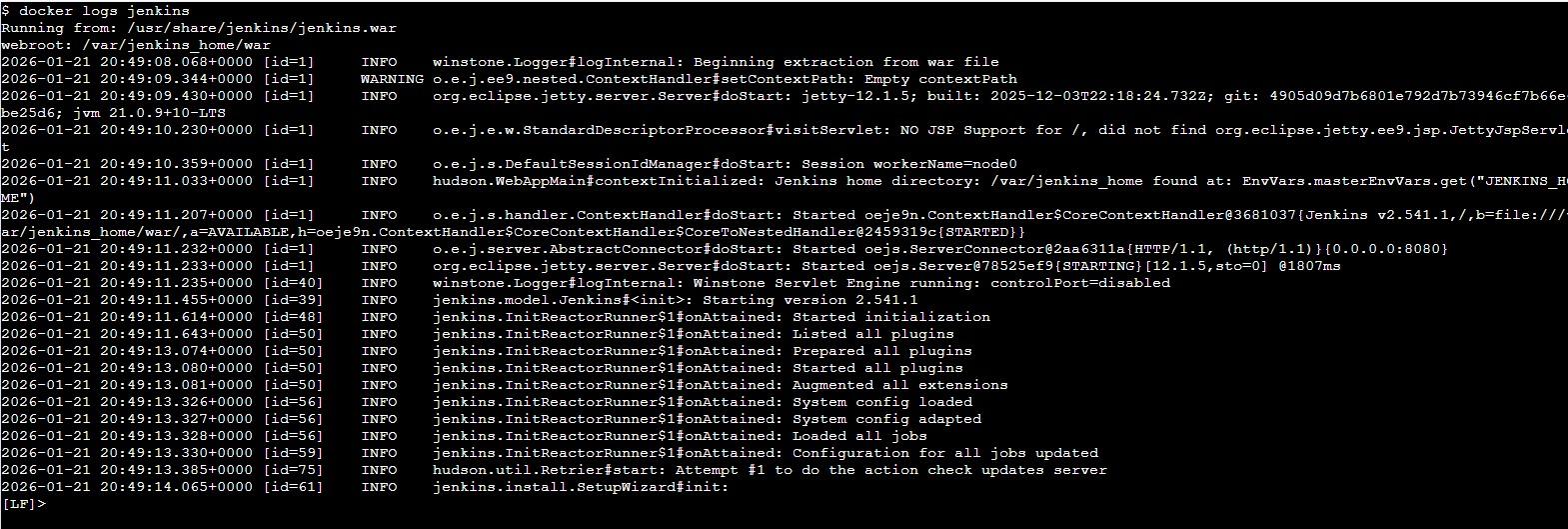


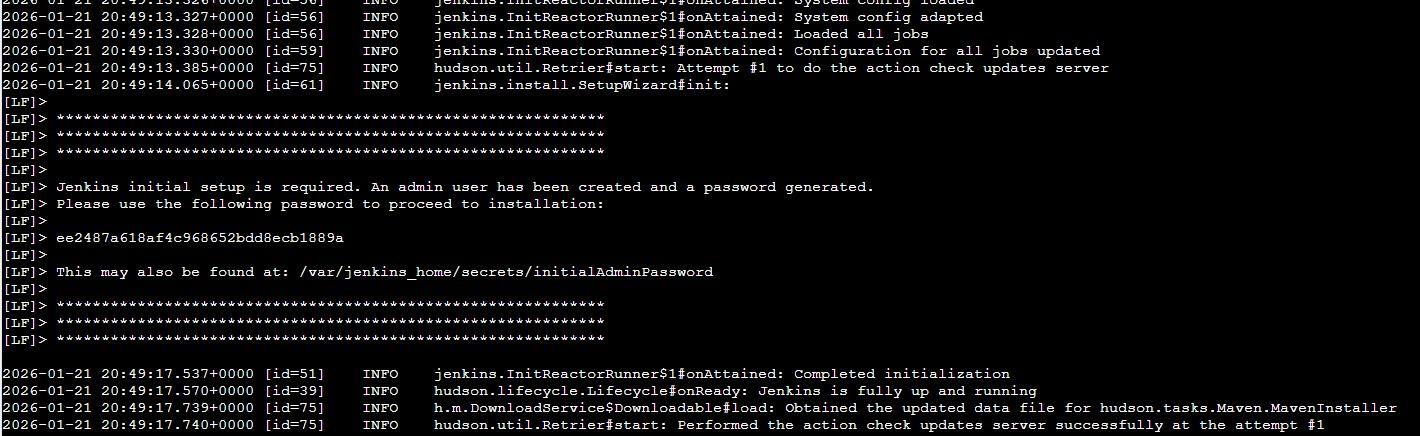
שקף 20:

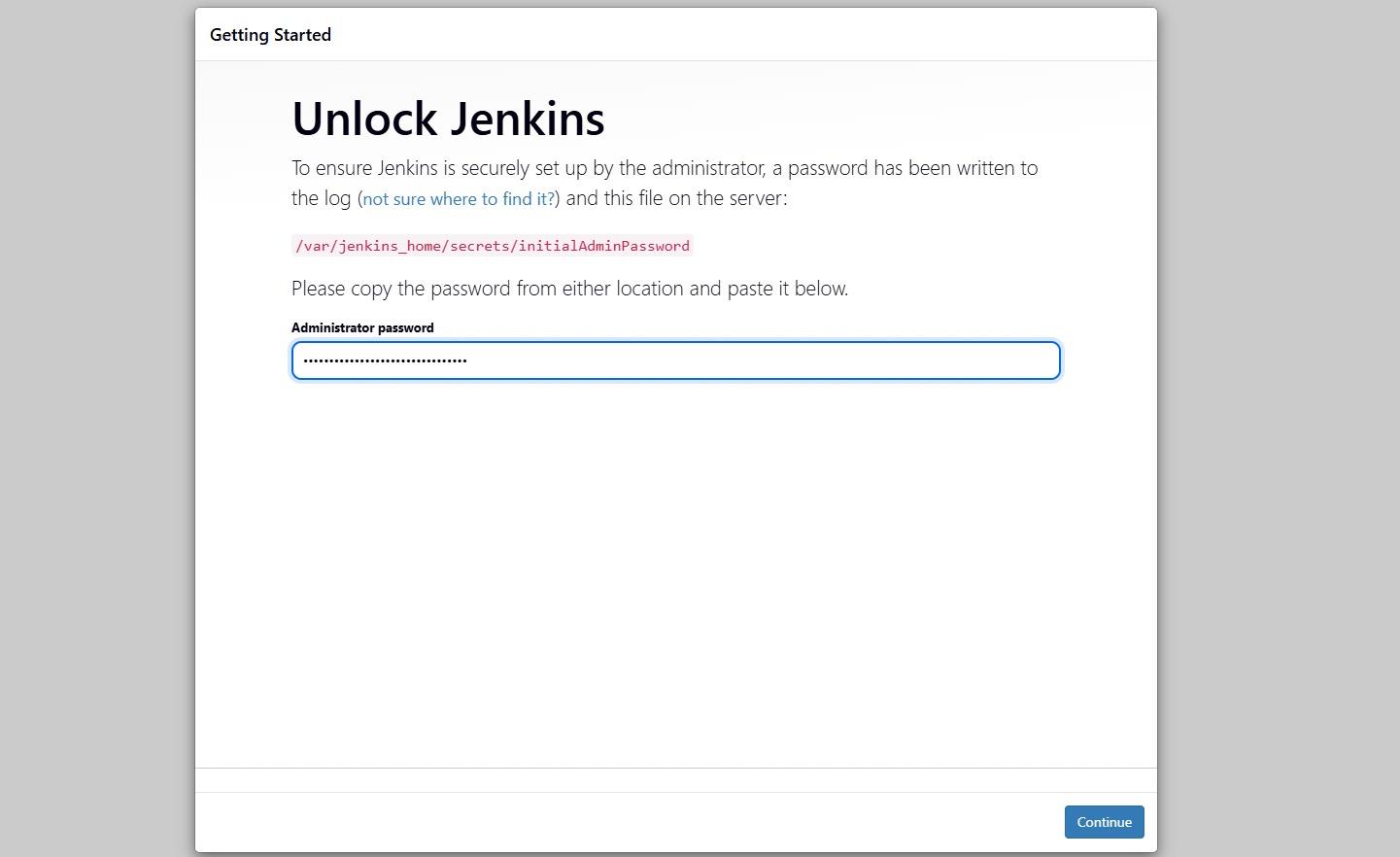


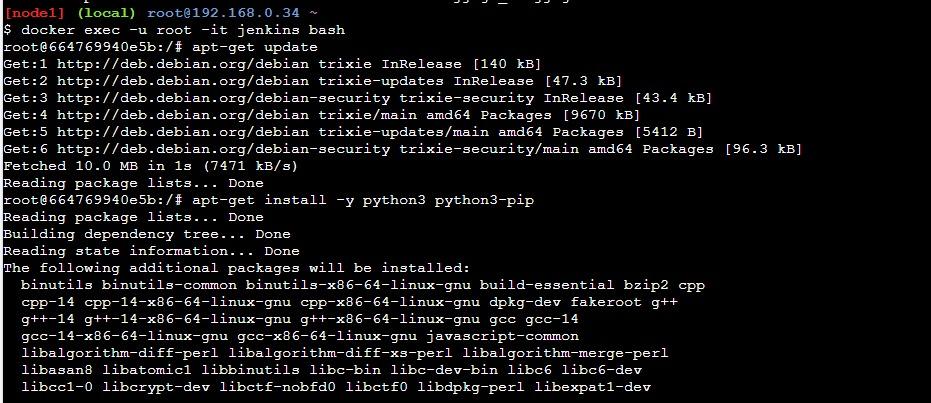


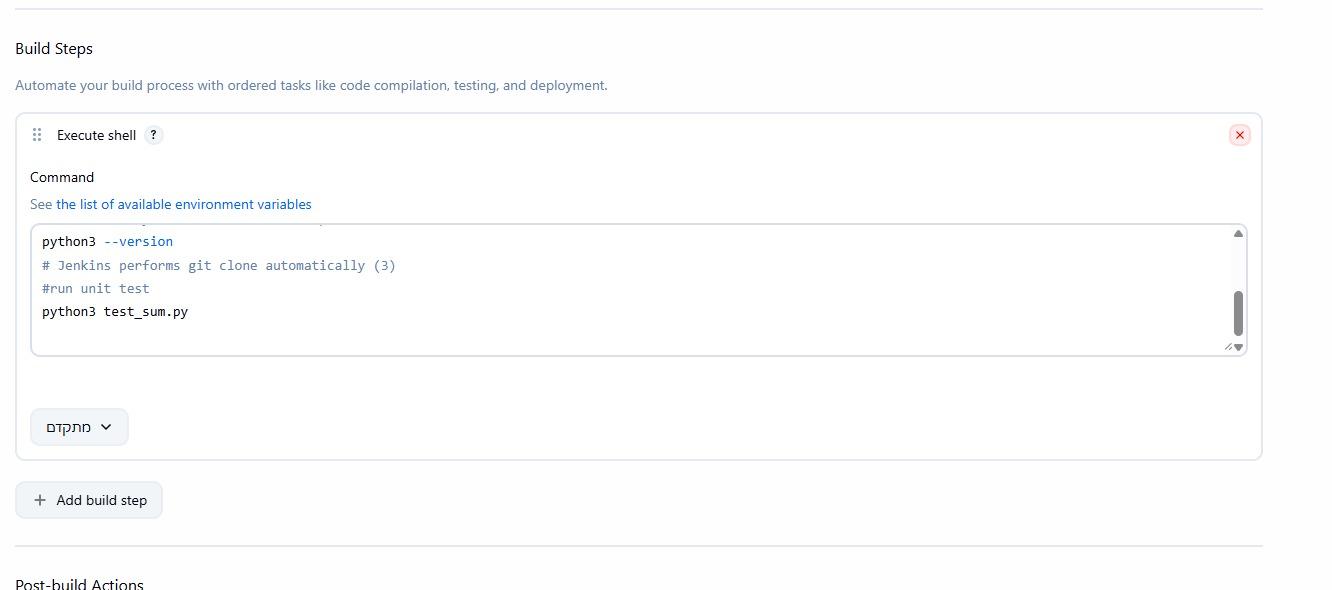
זה יביא לנו את המפתח להתחברות ל jenkins

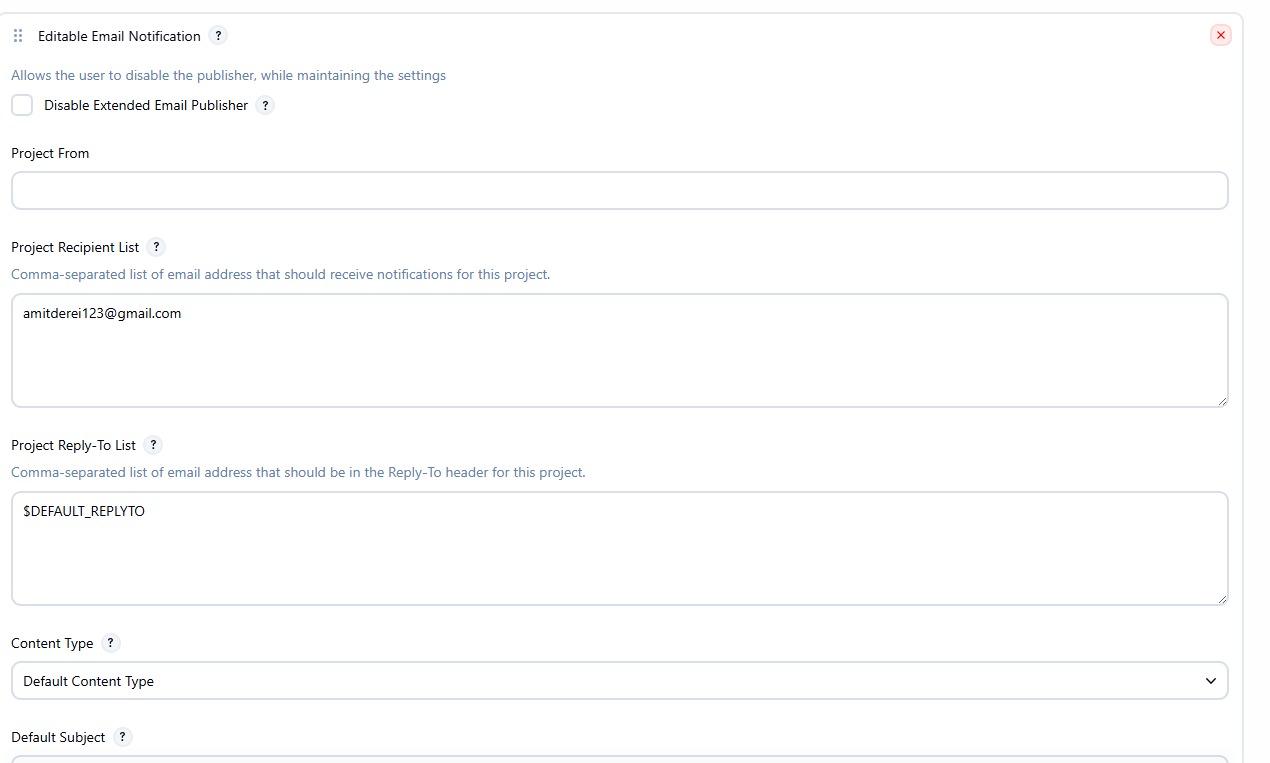


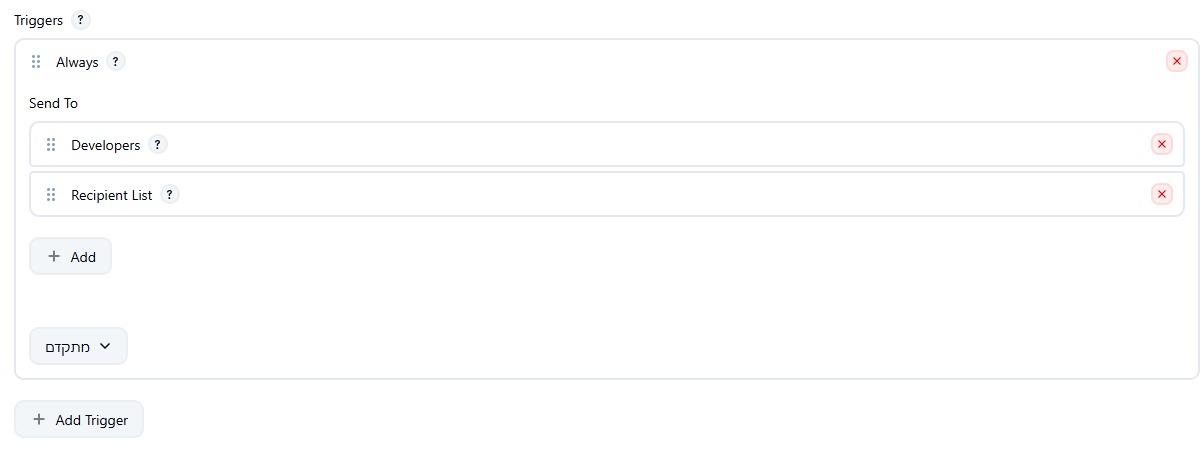


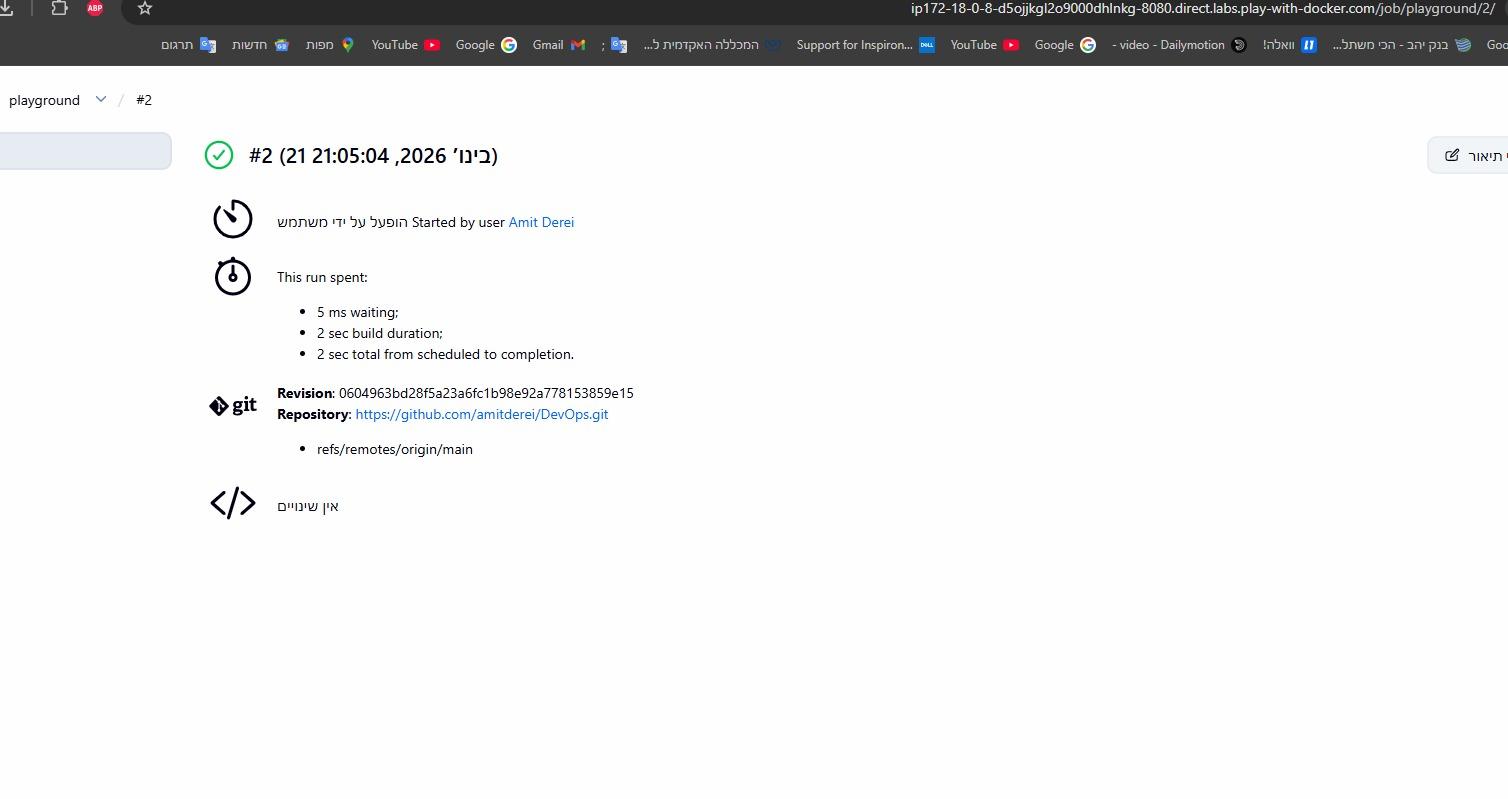


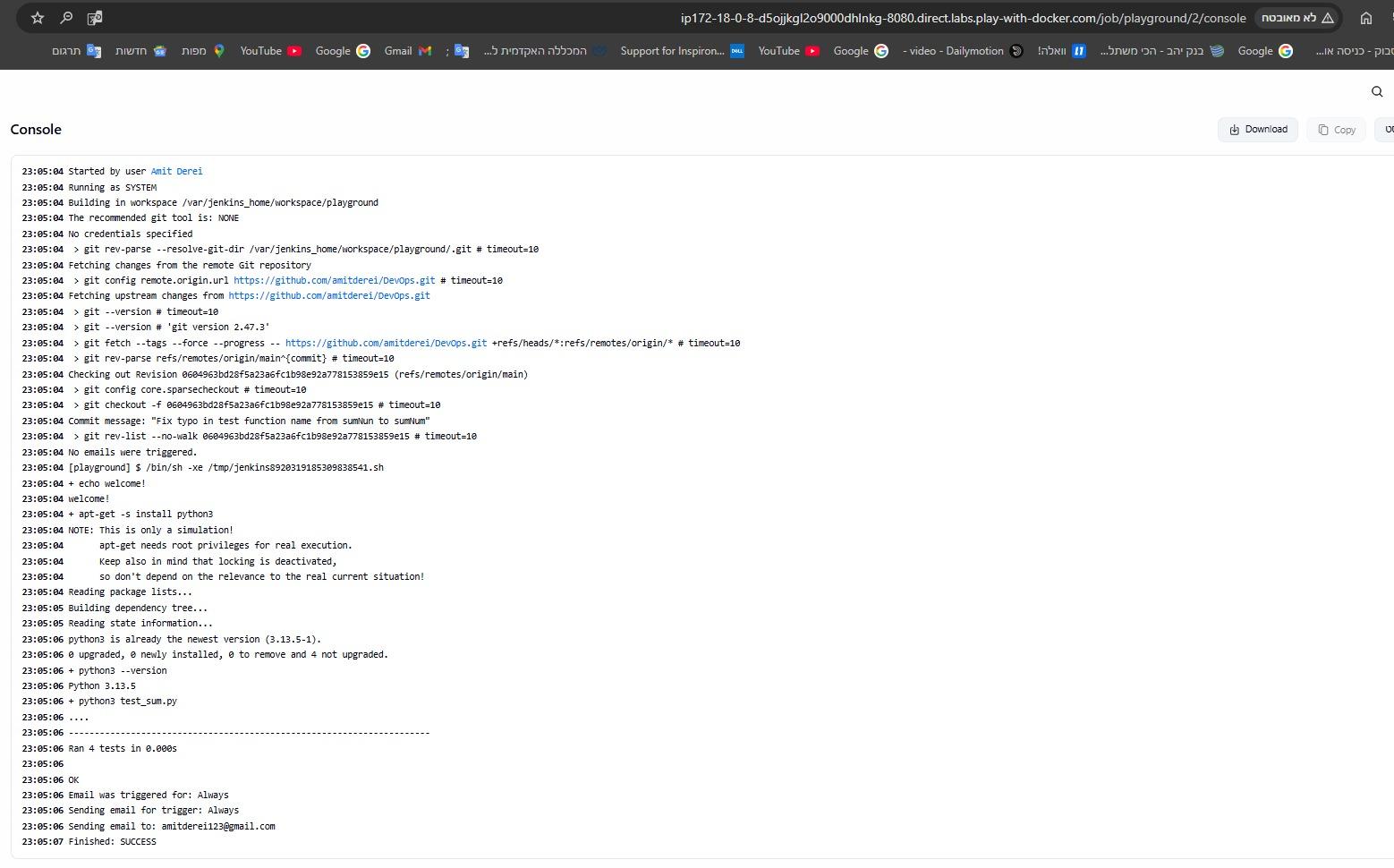


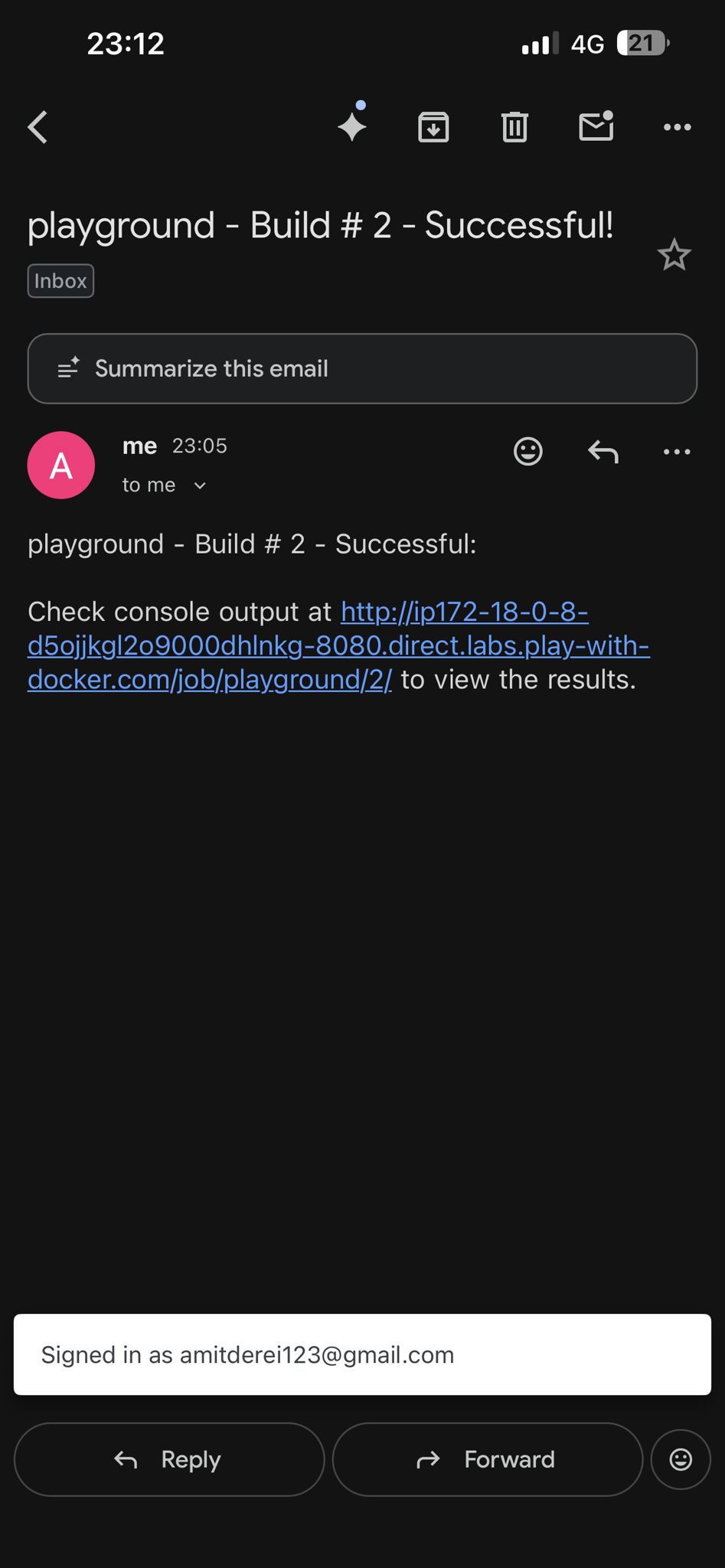












הרצאה 7. Kubernetes:

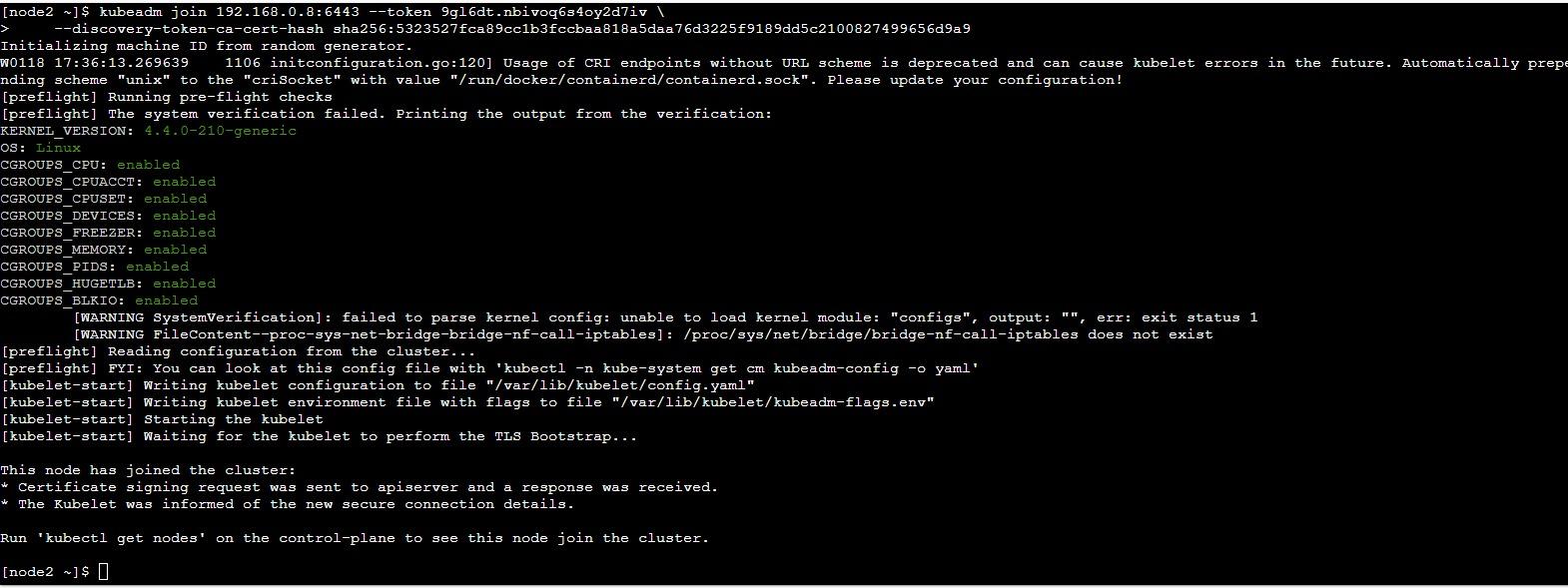
שקופית 31:

אתחלנו לפי מה שרשום היה ב1 initilize cluster networking ואז התחלנו לייצר את הnodes

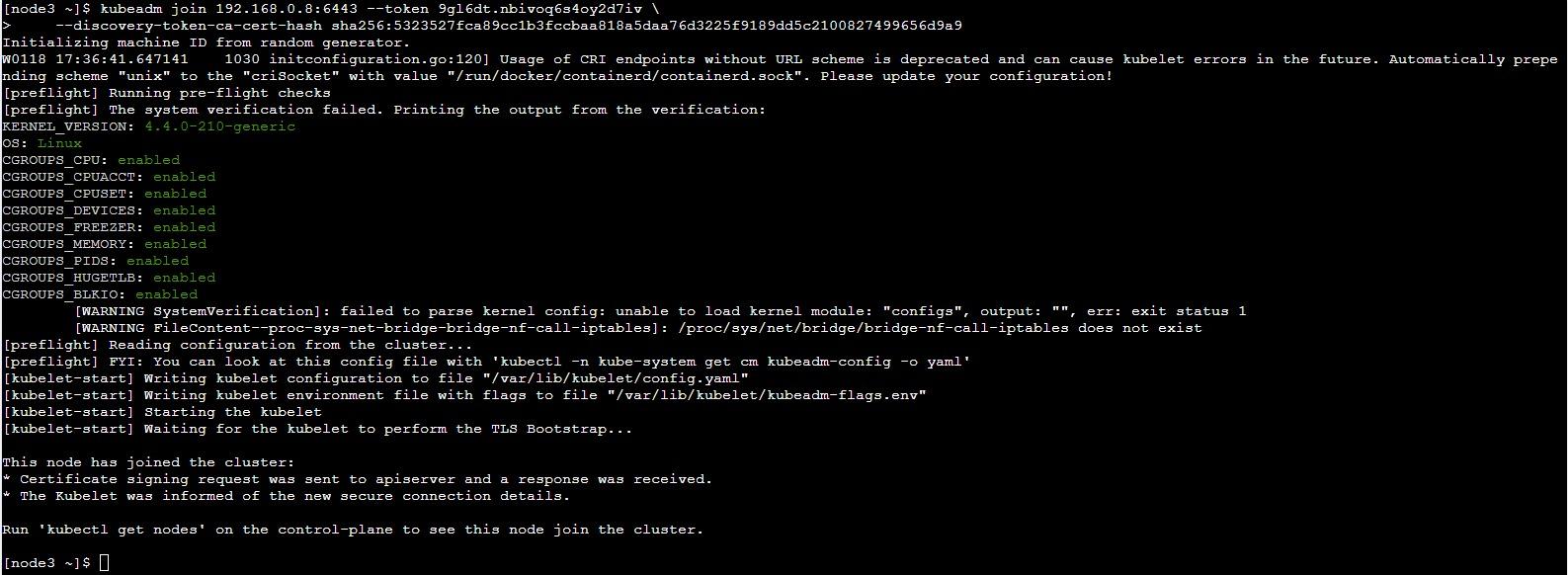
זה מאתחל הגדרות רשת נחוצות



NODE 2

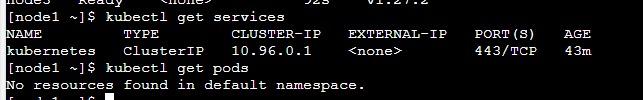


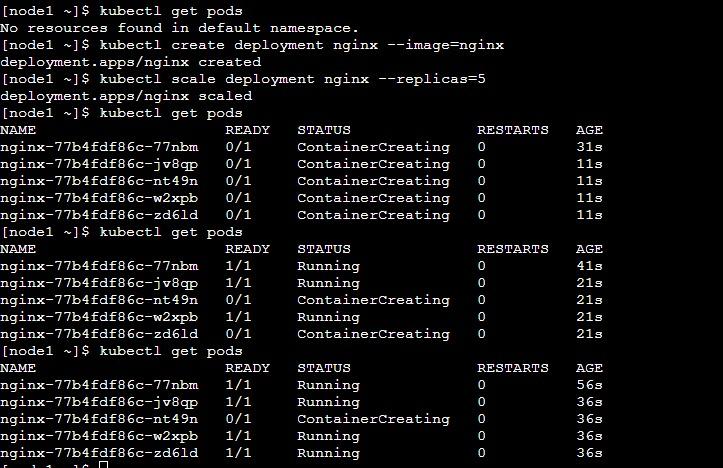
NODE 3

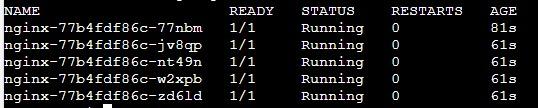


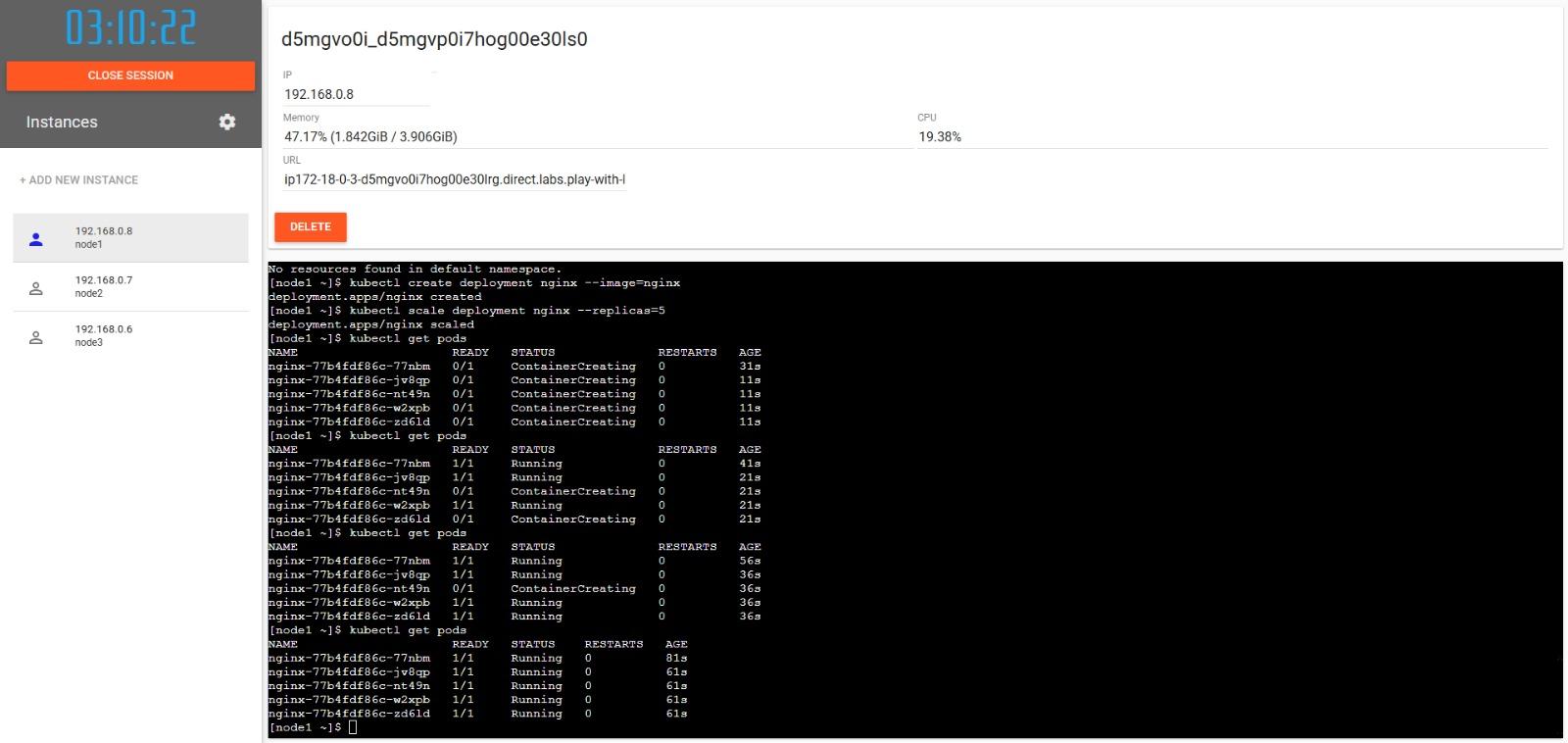


תמונה שמתארת את הcluster שלנו עם 3 nodes, הראשון מאסטר ו2 בנים workers - אימות הצמתים הפעילים





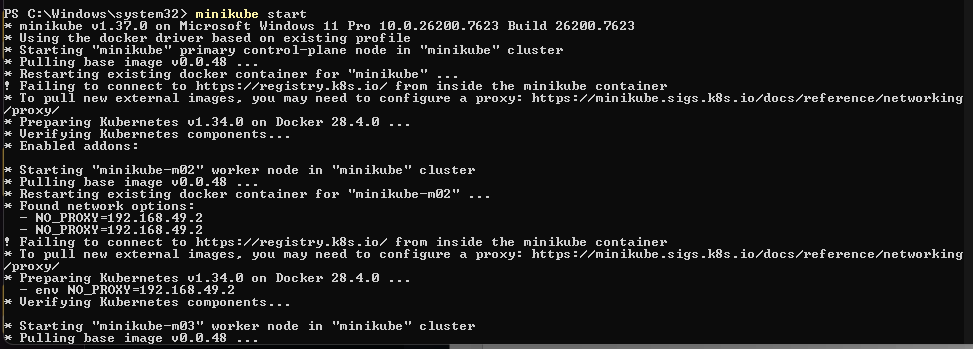
****

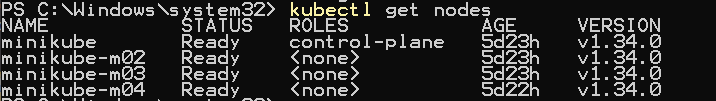
****

שקף 50 + 51: התחלנו לבצע לפי המצגת שלב אחר שלב, פקודה אחרי פקודה ולבסוף נתקלנו בבעיה של no running pod for service hello-minikube כדלהלן, למרות שעקבנו אחרי הפקודות וכן נמצא הservice, לא הצליח להריצו. מצורפות תמונות להמחשת השלבים שעשינו בתהליך.

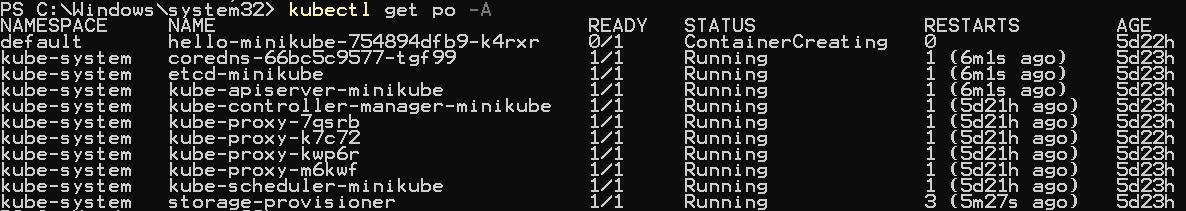
לאחר שיחה טלאפונית יחד איתך, אמרת לנו לתעד עד אותו הרגע:

התחלנו להריץ את minikube





אימות מצב ה-Minikube ושימוש ב-kubectl get po -A לצפייה בכל ה-Pods שרצים בכל מרחבי השמות



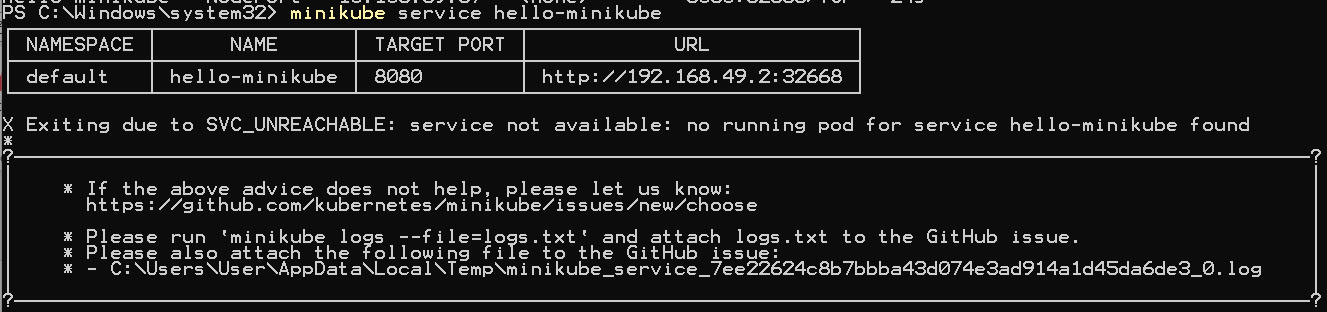
יצירת דיפלוימנט וחשיפה שלו כsrvice מסוג של פורט 8080





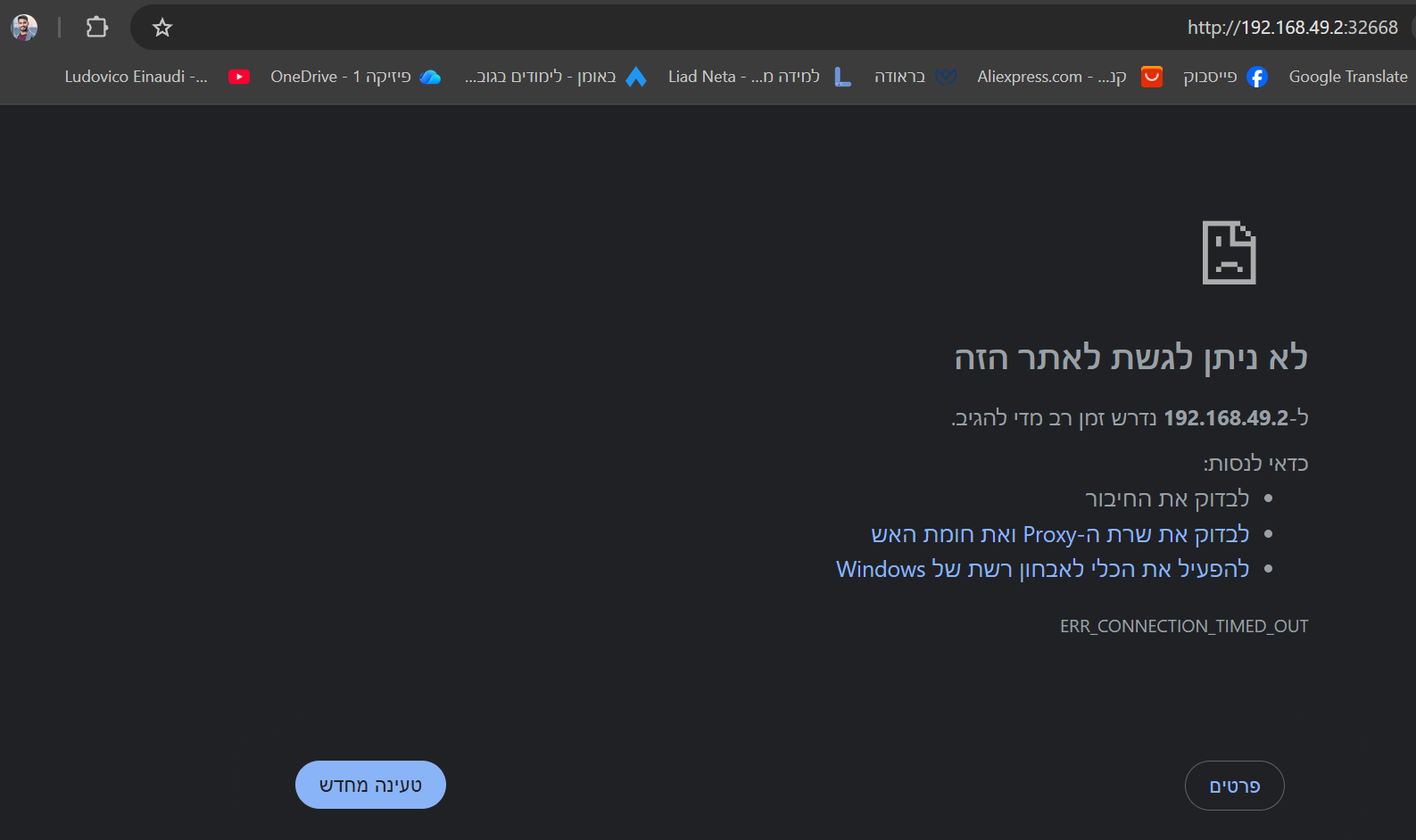


גישה לservice - פותח אוטומטית את הדפדפן בכתובת השירות שנוצר





נכנסו לפלט ואין טעינה של העמוד בHTTP בדפדפן..



יש לציין שגם כשנכנסתי דרך local host עם הפורט 8080, התקבלה אותה השגיאה, ולא נטען לי הדפדפן כמו שציפינו.