

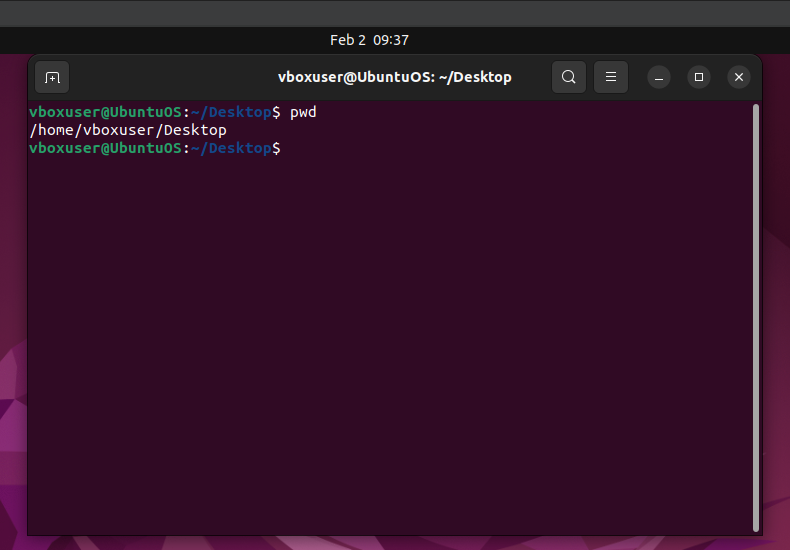
מגישים :

נדב ראובנס – 315819623

רון סיסו – 316398460

<https://github.com/RonSisso/DevOpsProject>

הרצאה 1 – Introduction :



הפקודה pwd (print working directory), כפי שניתן להבין משמה, מדפיסה את הpath המלא של התיקייה הנוכחית בה אנו נמצאים (working directory).

הרצאה 2 – Linux :

שקף 10:

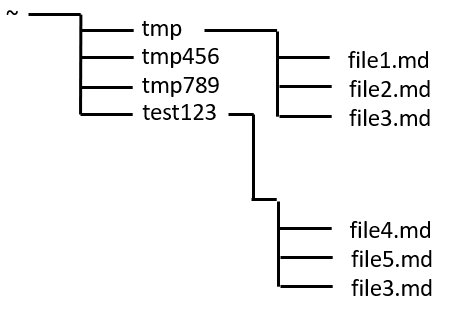
ניתן לראות מצילום המסך הקודם כי:

Username = vboxuser

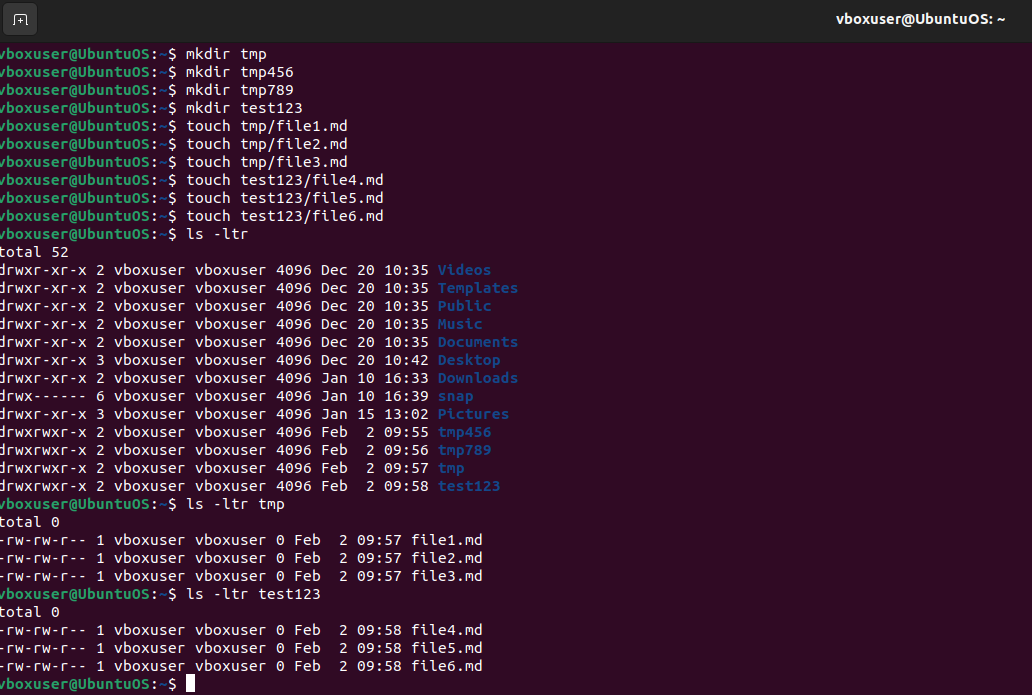
Machine name = UbuntuOS

Current directory = home/vboxuser/Desktop

שקף 26:



ניצור את המבנה הבא תחת הroot folder :



באמצעות פקודת mkdir יצרנו את כל התיקיות במבנה. לאחר מכן באמצעות פקודת touch יצרנו את הקבצים של המבנה בpath המתאים.

לבסוף באמצעות ls הראנו את רשימת הקבצים והתיקיות כדי לבדוק כי אכן נוצר המבנה הדרוש.

שקף 35:

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטיראשית נבצע את הפקודות ln,cat, man כפי שהן מופיעות במצגת :

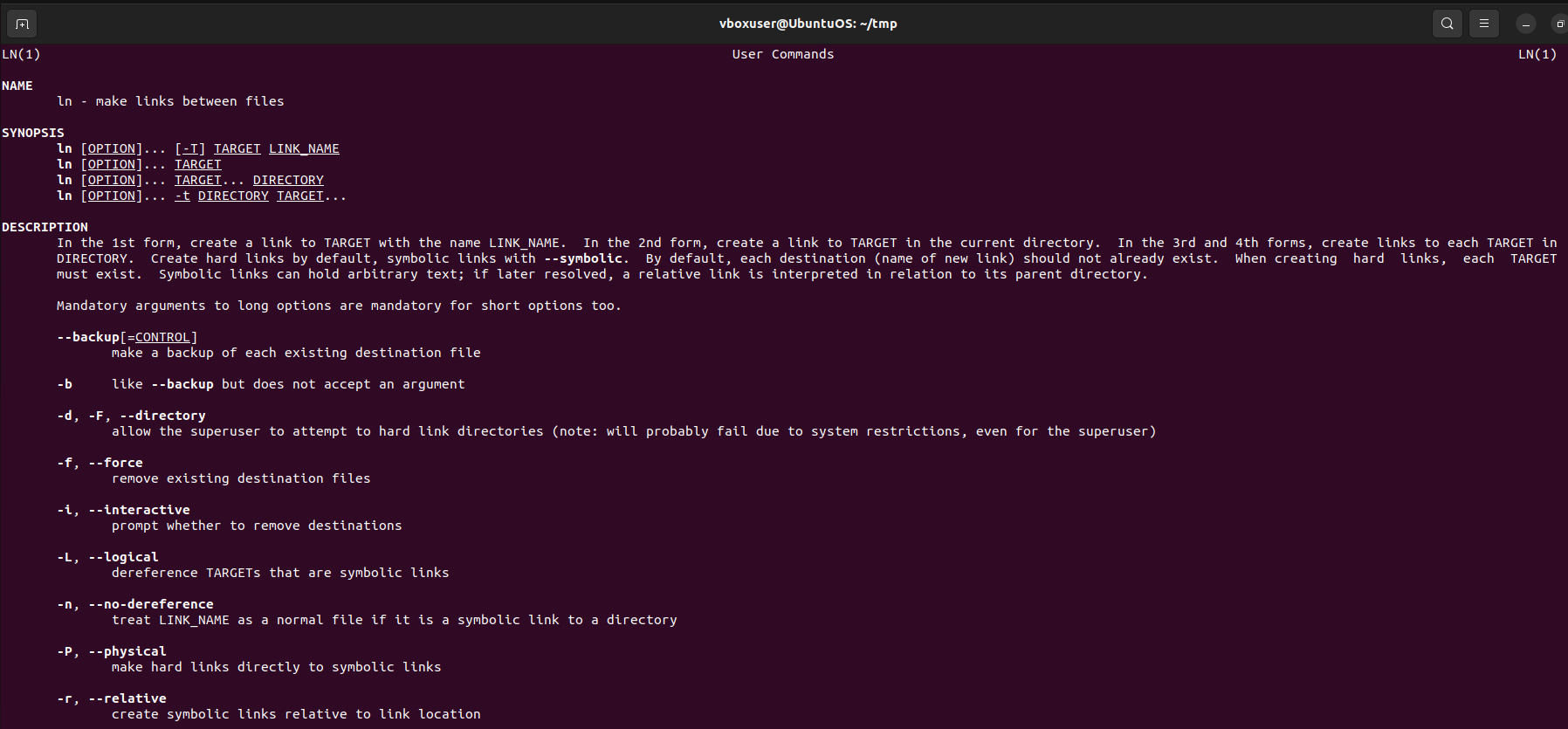
תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

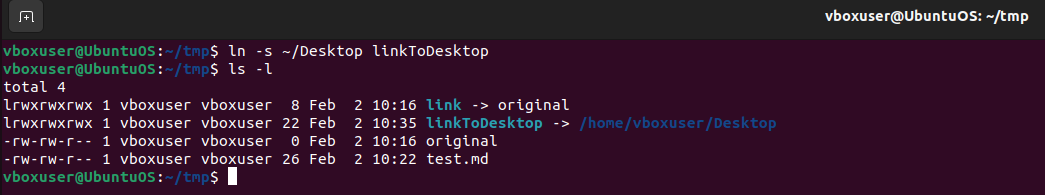
התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטינבצע הדפסת documentation של הפקודות ln וmv באמצעות הפקודה man:

תמונה שמכילה טקסט, צילום מסך

התיאור נוצר באופן אוטומטי

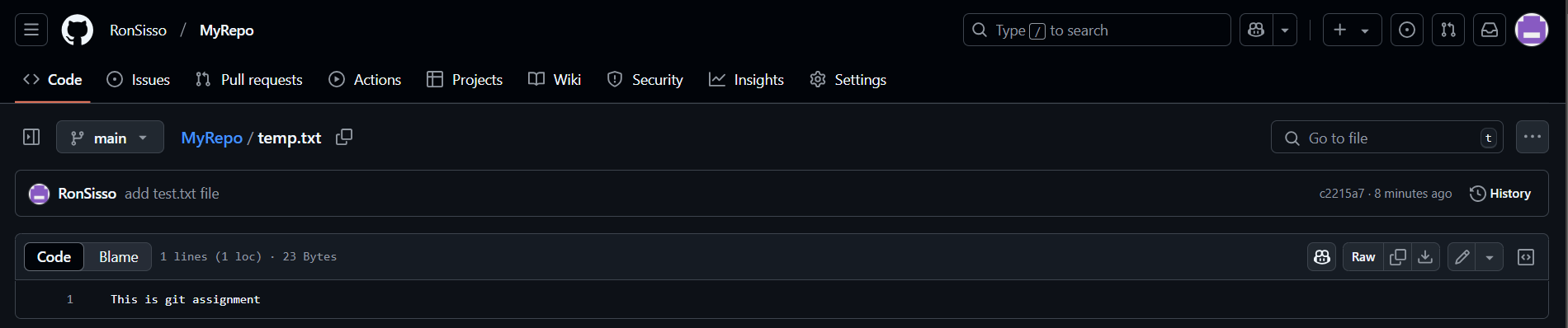
נבצע soft link לתיקיית הDesktop :

הרצאה 3 – Git :

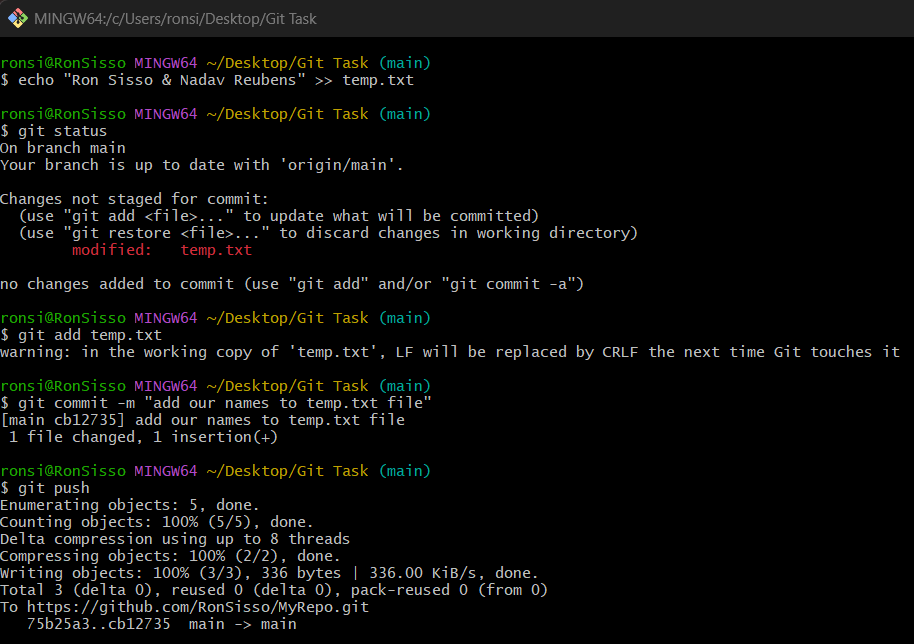
נבצע יצירת repository חדש בgithub ולאחר מכן נבצע CRUD כלומר

Create -> Read -> Update -> Delete על קובץ בשם temp.txt



נבצע קריאה של הקובץ בgithub :

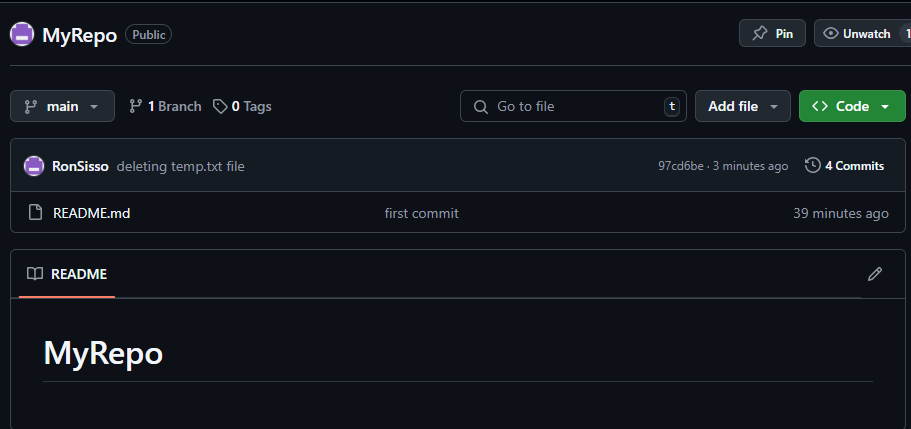
נבצע update על ידי הוספת תוכן נוסף לקובץ temp.txt. לאחר מכן נבצע commit וpush לrepository שלנו.

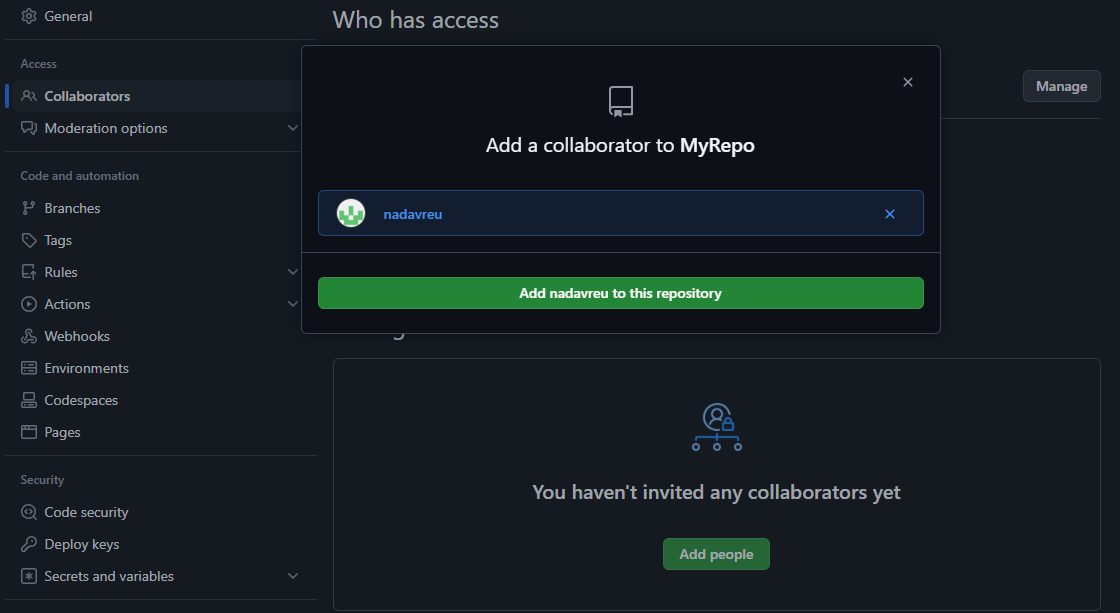


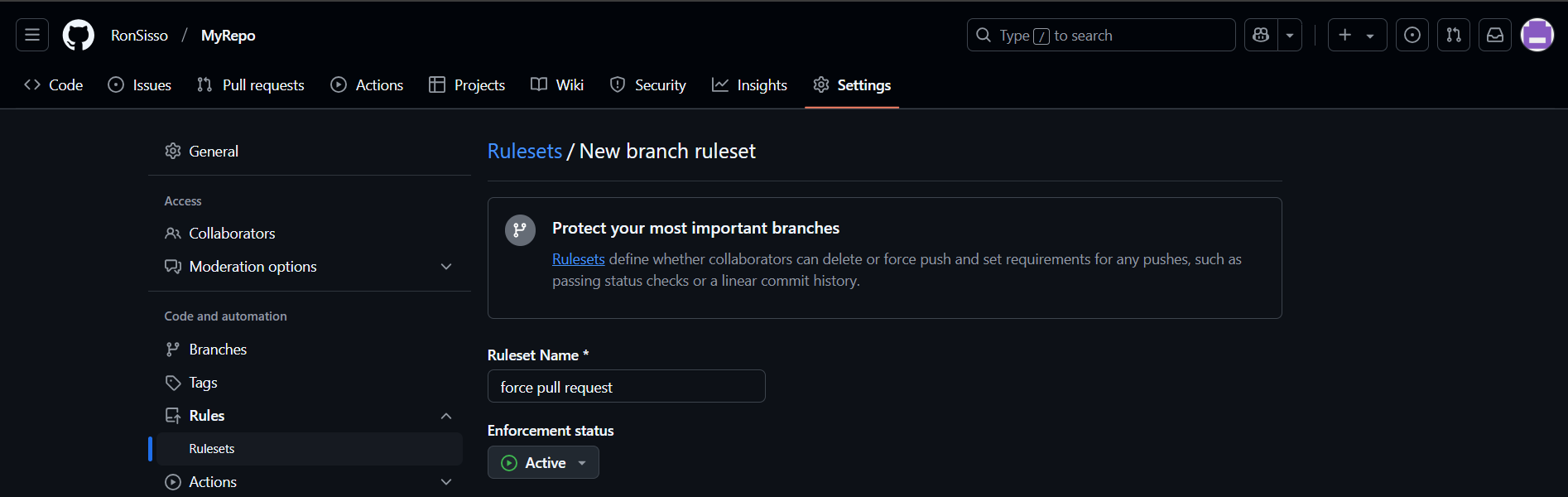
נבצע delete לקובץ temp.txt ע"י הפקודה git rm

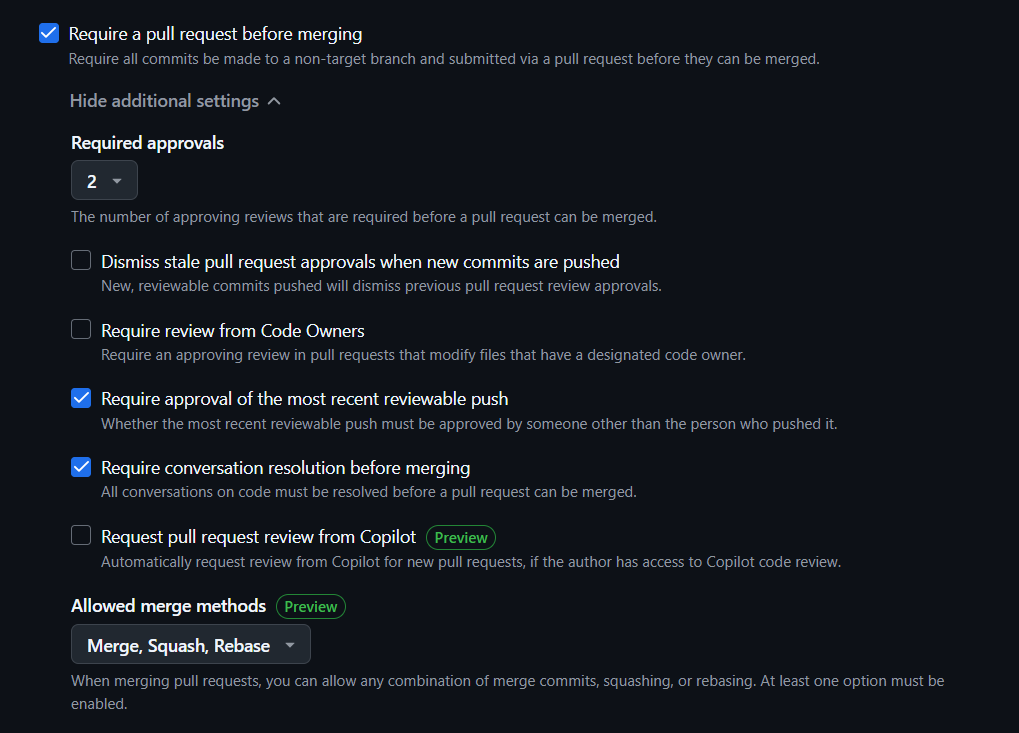
תמונה שמכילה טקסט, חשמל, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי



נוסיף את חבר הצוות השני אל הrepository:

נבצע configure לrepository עבור force pull request :



Dependabot הוא כלי מובנה ב-GitHub שמעדכן אוטומטית את ה-dependencies בפרויקט. הוא מזהה גרסאות ישנות, מסייע לשמור על קוד מאובטח ומציע עדכונים דרך pull requests.

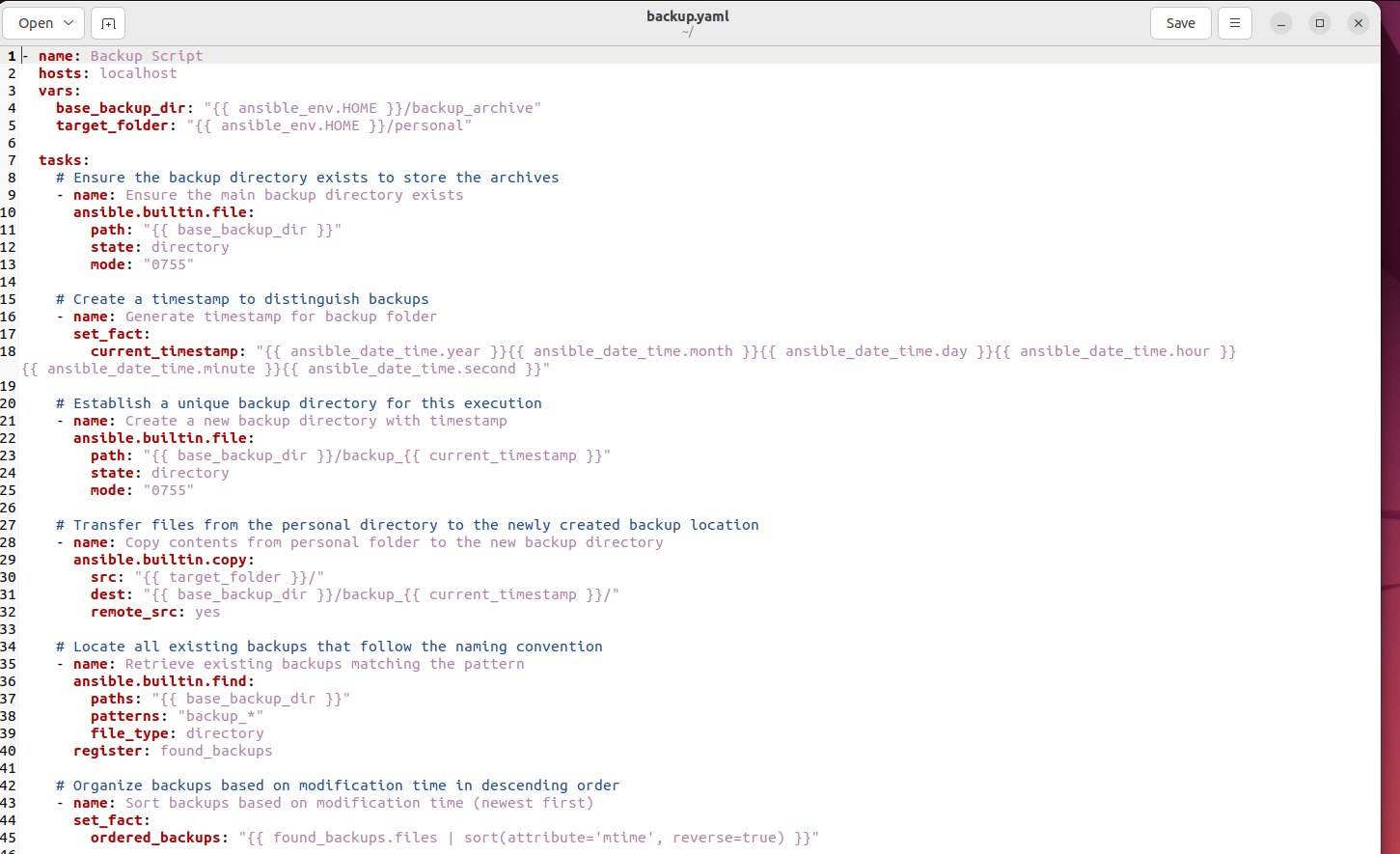
מה הצורך שלנו בDependabot ?

הוא שומר על הפרויקט שלנו מאובטח, מעודכן ויציב. הוא מזהה אוטומטית נקודות תורפה באבטחה ומציע עדכונים ל-dependencies, מה שחוסך זמן, מקטין סיכונים ומונע בעיות תאימות. במקום לעדכן חבילות ידנית, Dependabot עושה זאת אוטומטית ושומר על הפרויקט ברמה הגבוהה ביותר.

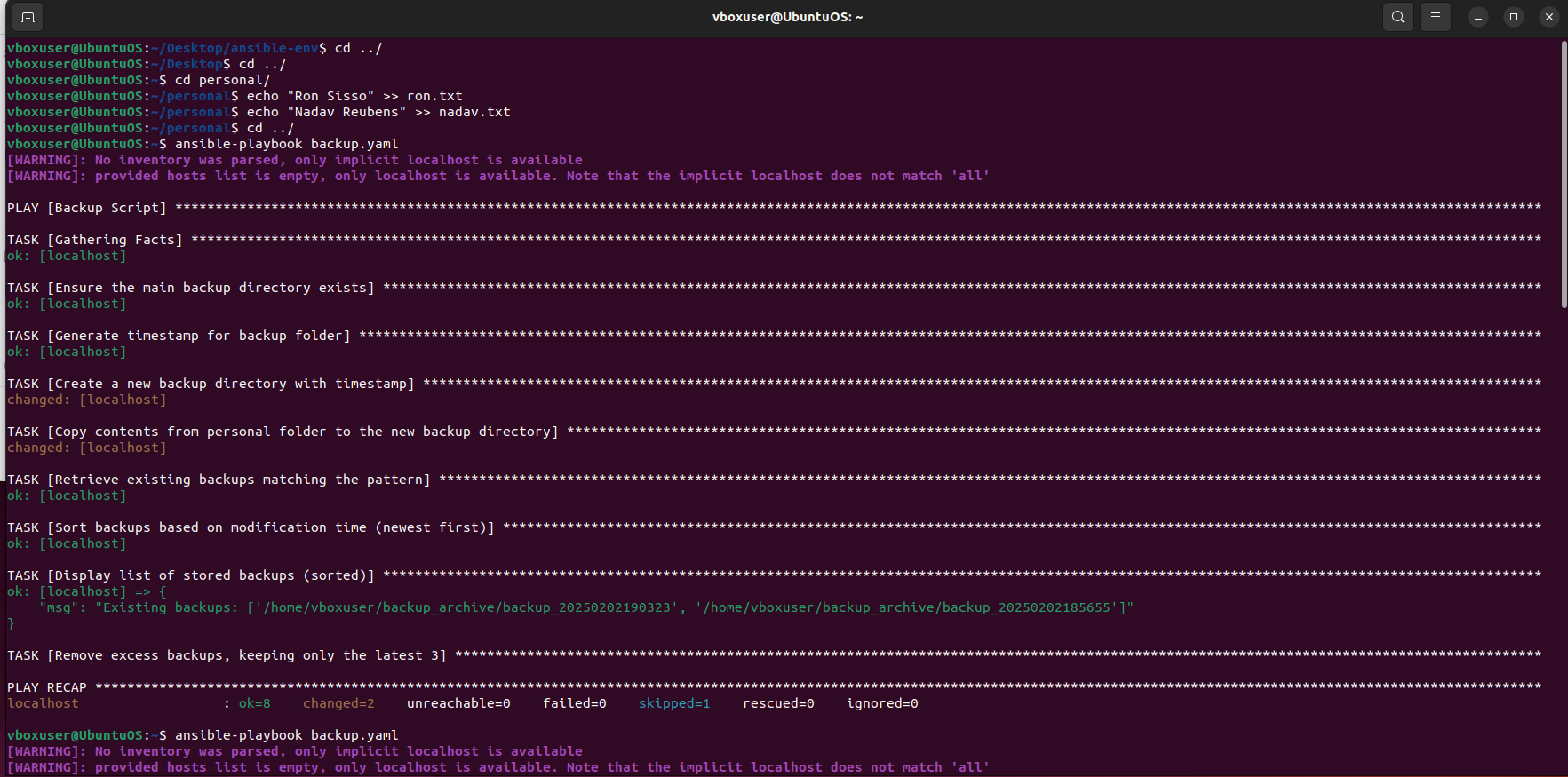
נשתמש ב Dependabot כאשר נרצה לוודא שהפרויקט שלנו מעודכן ובטוח מפני פרצות אבטחה. במיוחד, נשתמש בו כאשר GitHub מזהה פגיעות אבטחה ב dependenciesכאשר אנו רוצים לשמור על יציבות הקוד ולהימנע מבעיות תאימות, וכאשר הפרויקט שלנו כולל CI/CD או רץ בסביבת production שבה עדכונים שוטפים הם קריטיים. בנוסף Dependabot שימושי כאשר אנו רוצים לחסוך זמן על ידי עדכון אוטומטי של החבילות במקום תחזוקה ידנית.

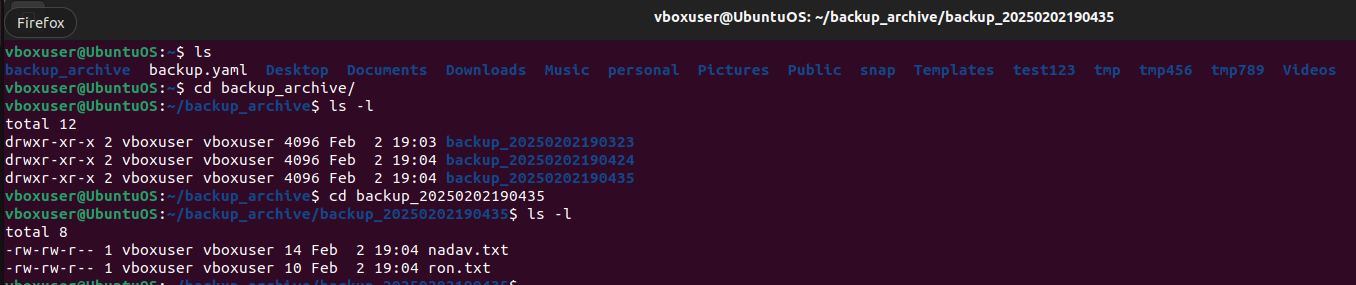
Dependabot חיוני לכל פרויקט כדי לשמור על ביטחון, יציבות ותחזוקה קלה.

תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטיהרצאה 4 – Ansible :

\* הקוד מצורף בצורה ברורה יותר כקובץ yaml בתיקיית הgit של הפרויקט שלנו.

הרצה מהterminal :



**הרצאה 5: Docker**

**שקף 29:**

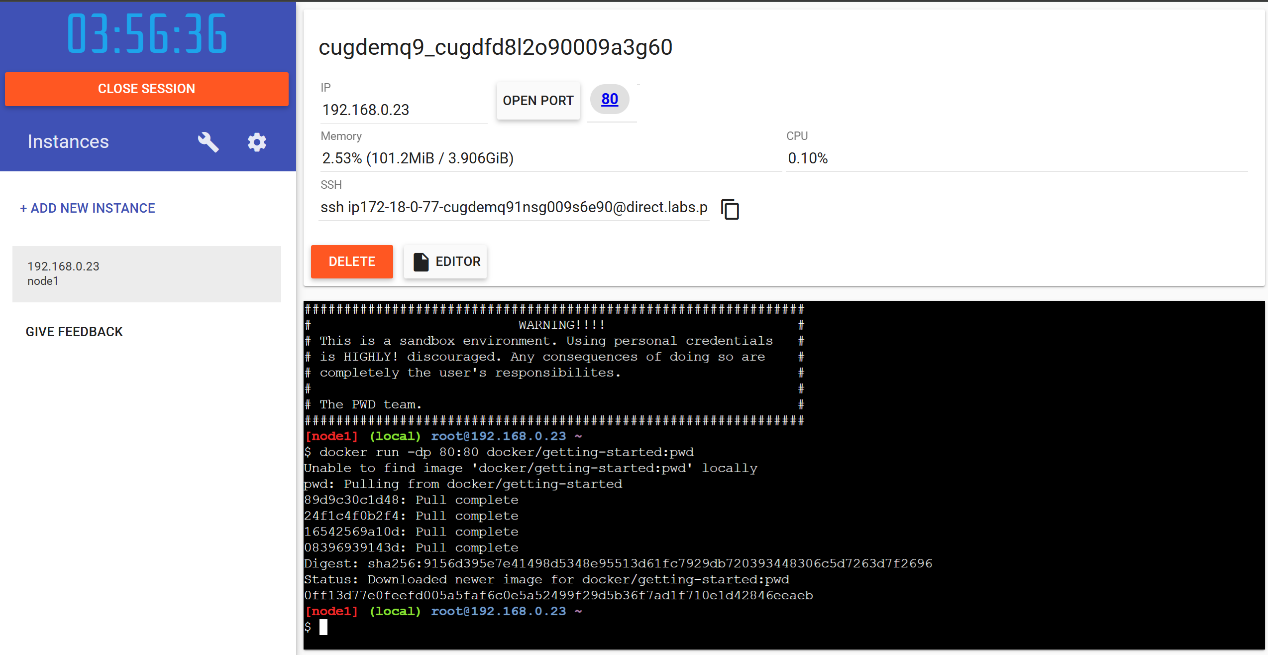
**פירוט המשימה:**

* כניסה לאתר <https://labs.play-with-docker.com/>

לחיצה על add new instance

הרצת הפקודות הבאות:

* % docker run -dp 80:80 docker/getting-started:pwd
* Docker run: Docker command to create and start new container from image
* -dp
* -d: detached mode. Runs the container in background
* -p: publish ports. Mapping a port on the host to a port in the container



מה מתרחש:

חיפוש התמונה מקומית

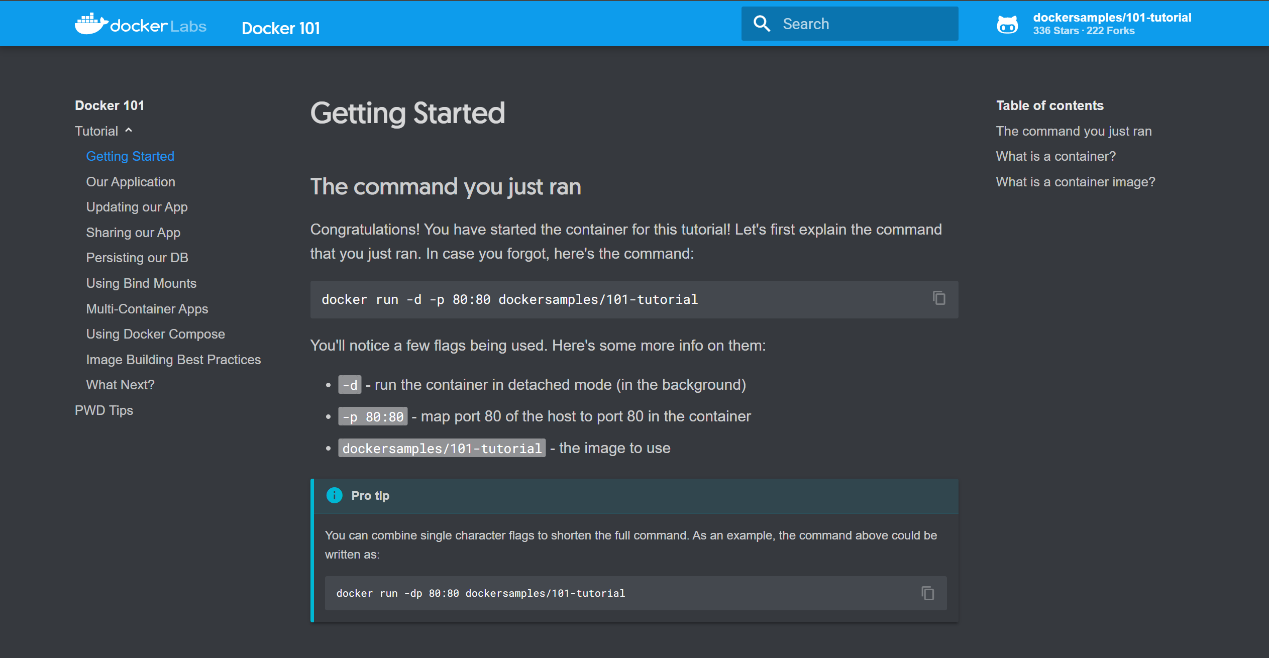
משיכת התמונה מהמאגר של Docker

התקדמות הורדת חלקי התמונה

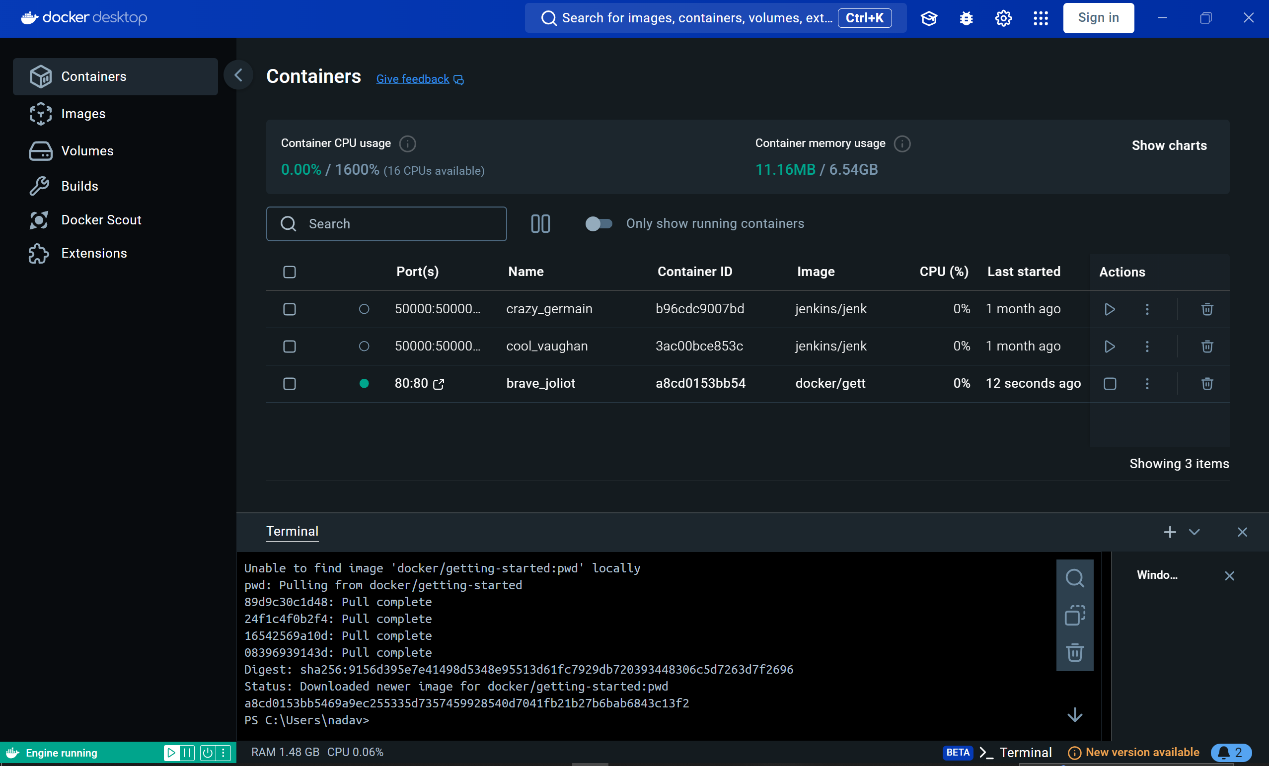
אימות התמונה לאחר הורדה

הורדה הושלמה בהצלחה

יצירת קונטיינר חדש והרצה

אפשר להשתמש בdocker ps כדי לראות את הקונטיינרים שרצים 

**שקף 33:**



נריץ את הפקודה docker run -dp 80:80 docker/getting-started:pwd מדוקר דסקטופ

נפתח את הקישור <http://localhost> ונקבל

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

**שקף 45:**

ליצירת קונטיינר Docker שמריץ סקריפט פייתון פשוט, תחילה נצור קובץ בשם app.py ונוסיף אליו את השורה print("Hello, Docker!"), ולאחר מכן נשמור את הקובץ. כעת, נצור קובץ נוסף בשם Dockerfile באותה תיקייה, וכתוב בו את ההוראות הבאות FROM python:3.10-slim לשימוש בתמונה בסיסית של פייתון, COPY app.py /app.py כדי להעתיק את הקובץ לתוך הקונטיינר, ו- CMD ["python", "/app.py"] כדי לקבוע את הפקודה שתופעל בעת הרצת הקונטיינר. לאחר מכן, יש לבנות את התמונה באמצעות הפקודה docker build -t hello-docker . ולוודא שהתמונה נבנתה בהצלחה.

כדי להריץ את הקונטיינר ולהציג את הפלט, השתמש בפקודה docker run hello-docker, ואם הכול תקין, יוצג הפלט "Hello, Docker!".

קוד app.py:

תמונה שמכילה טקסט, צילום מסך, גופן

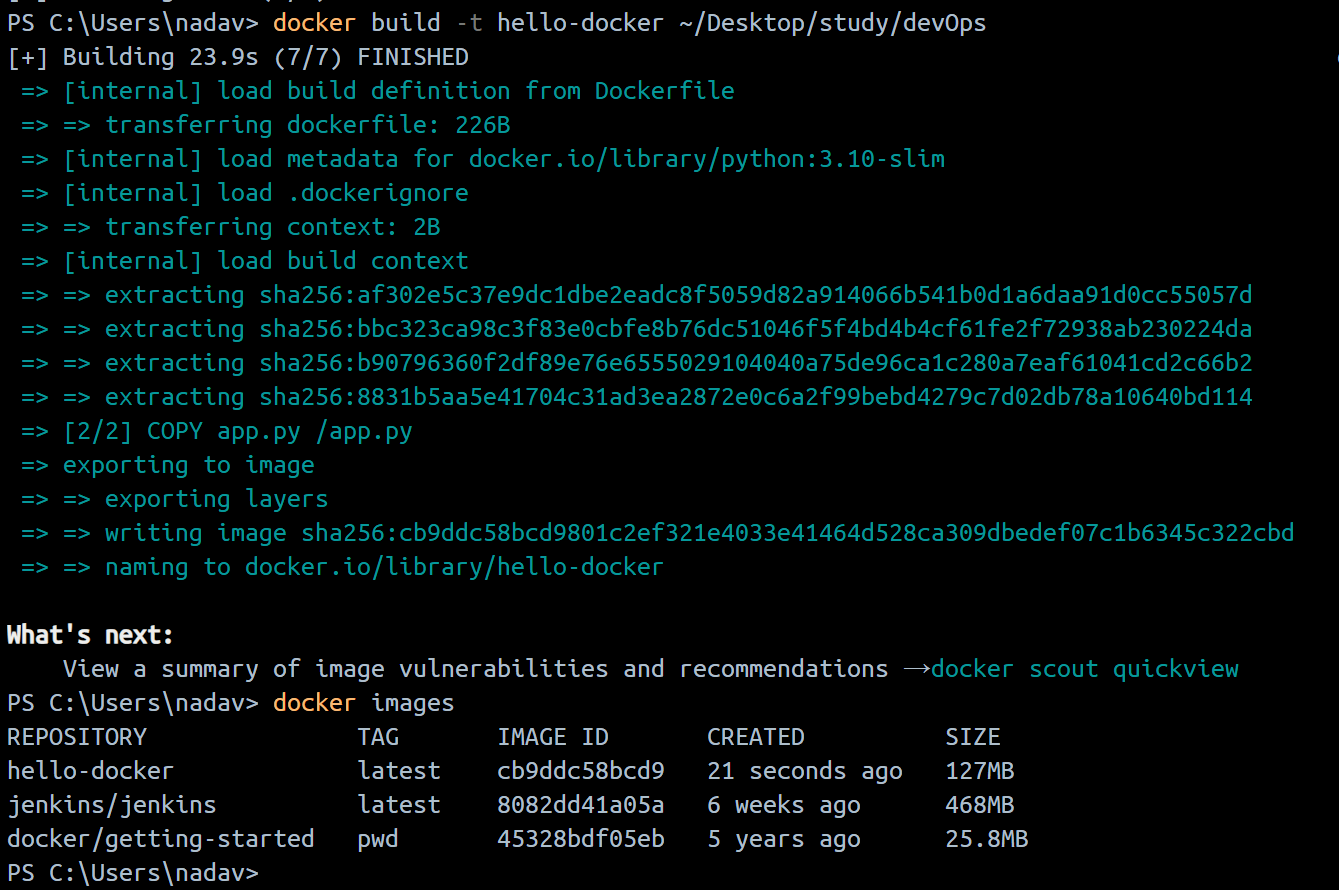
התיאור נוצר באופן אוטומטי

קובץ Docker:

תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

התיאור נוצר באופן אוטומטי

Image build:



הרצת הקונטיינר:

תמונה שמכילה טקסט, גופן, צילום מסך

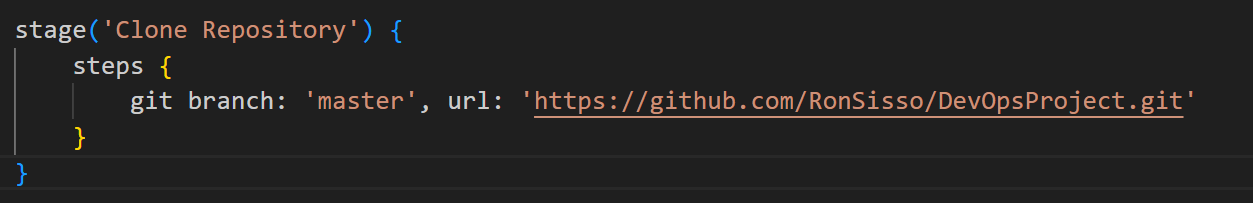
התיאור נוצר באופן אוטומטי

**הרצאה 6:**

**שקף 19:**

* Create Jenkinks Job, which do the following:
* 1. Print welcome message
* תמונה שמכילה טקסט, צילום מסך, גופן

  התיאור נוצר באופן אוטומטי
* 2. Installs python & verify python installation with “python --version"
* תמונה שמכילה טקסט, צילום מסך

  התיאור נוצר באופן אוטומטי
* 3. Check-out your repository from master branch
* 4. Runs unit-tests on your code
* תמונה שמכילה צילום מסך, גופן, קו, טקסט

  התיאור נוצר באופן אוטומטי
* 5. Sends email for every run result

תמונה שמכילה טקסט, צילום מסך, גופן, תוכנה

התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

**הרצאה 7:**

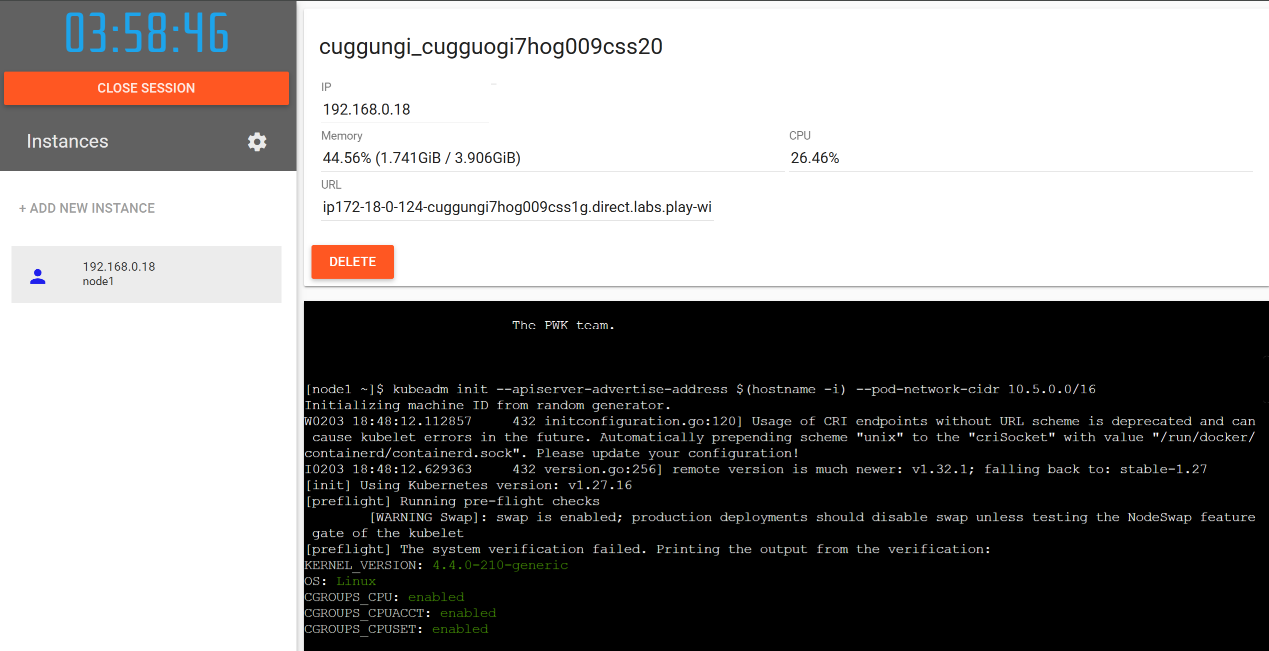
**שקף 31:**

ניצור את המאסטר נוד באמצעות הפקודה:

kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

configuration:

kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube-router/master/daemonset/kubeadm-kuberouter.yaml



תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

נצרף worker node :

kubeadm join 192.168.0.18:6443 --token 4m2kwf.xjxj687lx6dzvp2q \

--discovery-token-ca-cert-hash sha256:66228517792379e1af37dce716d568a40f0db0e600ec5e133bd90defc7ded485

פקודה זו ניתן למצוא בפלט של יצירת המאסטר נוד

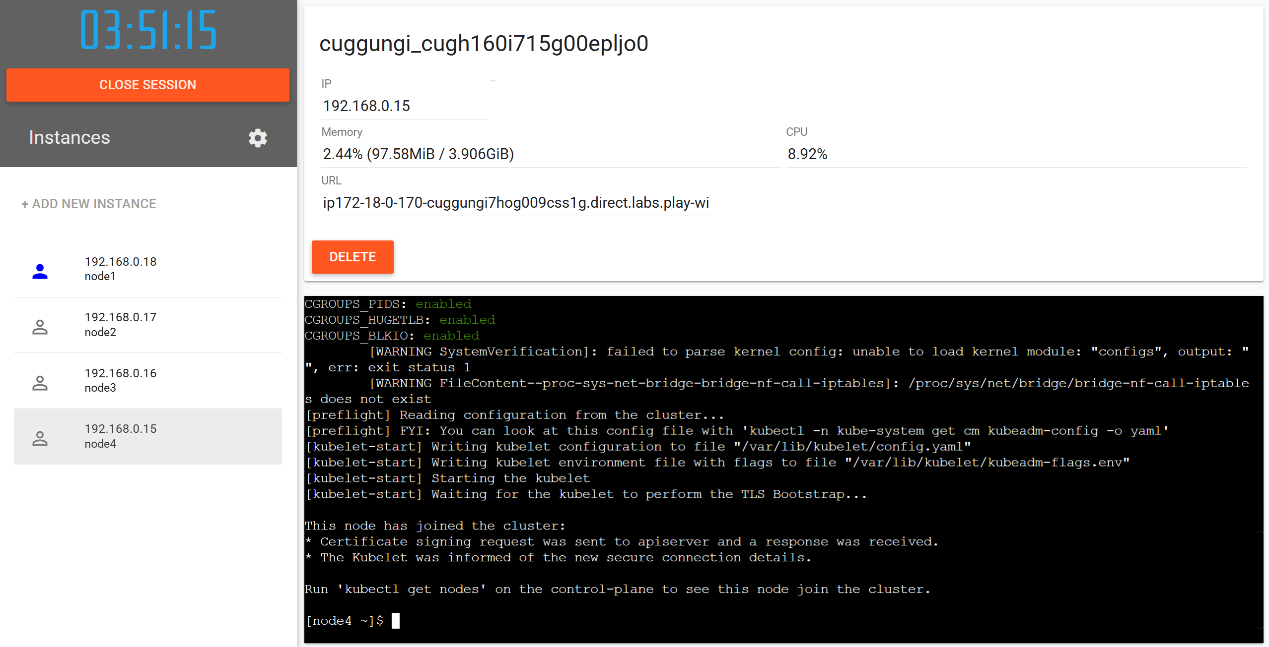
תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

נוסיף worker node שני

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

נוסיף worker node שלישי

נבדוק במאסטר נוד בעזרת:

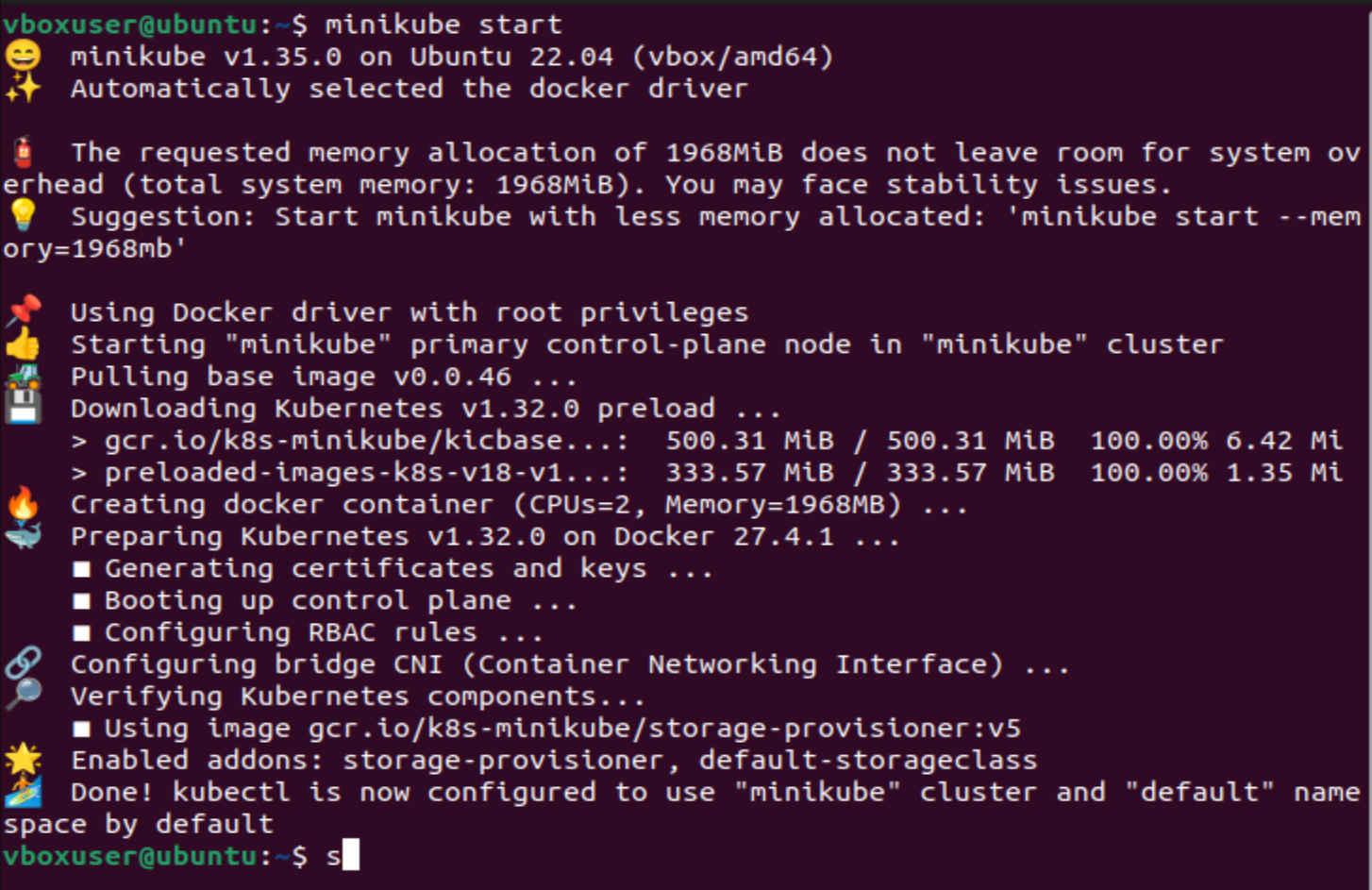
תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

**שקף 50:**

נרצה לבצע בדיקות POST וGET בPOSTMAN לכתובת <http://localhost:7080/>

נריץ minikube



נצור פריסת Kubernetes עם תמונת השרת

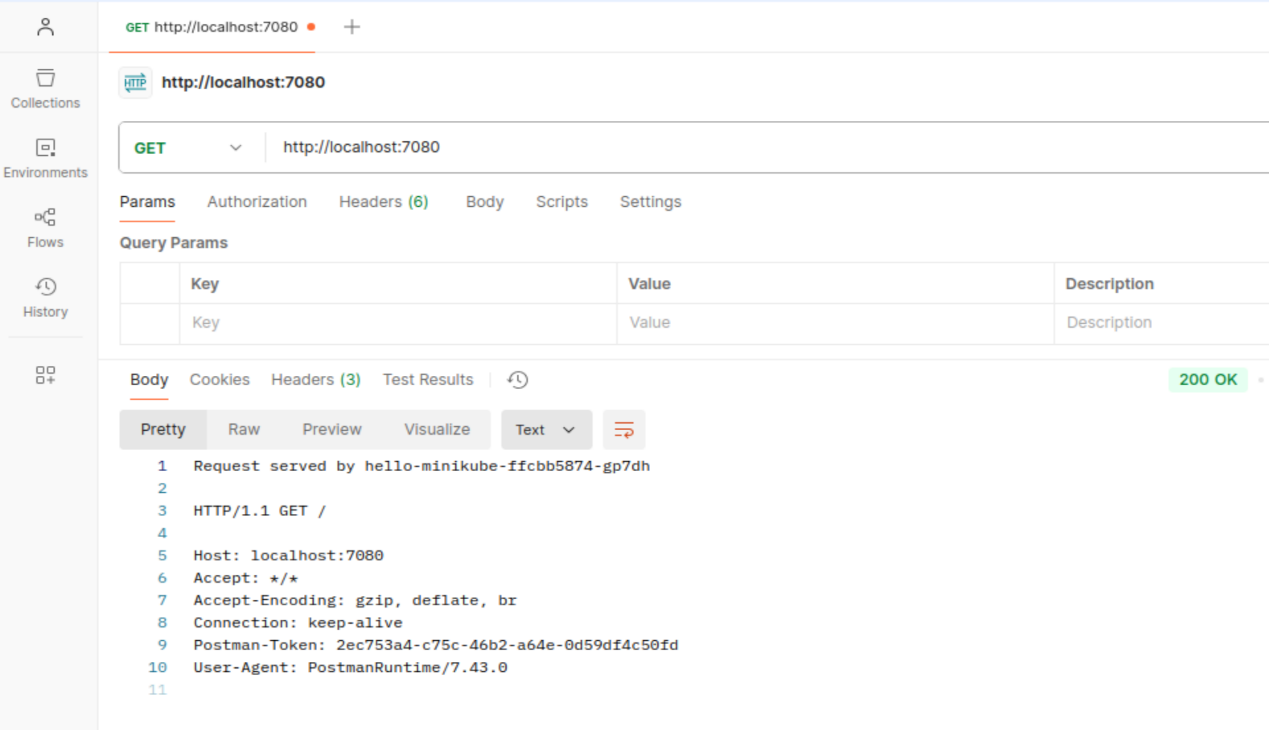
נחשוף את השירות עם NodePort בפורט 8080

ונבדוק פרטי השירות:

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

נבצע קריאת GET:



נבצע קריאת POST:

תמונה שמכילה טקסט, גופן, קו, מספר

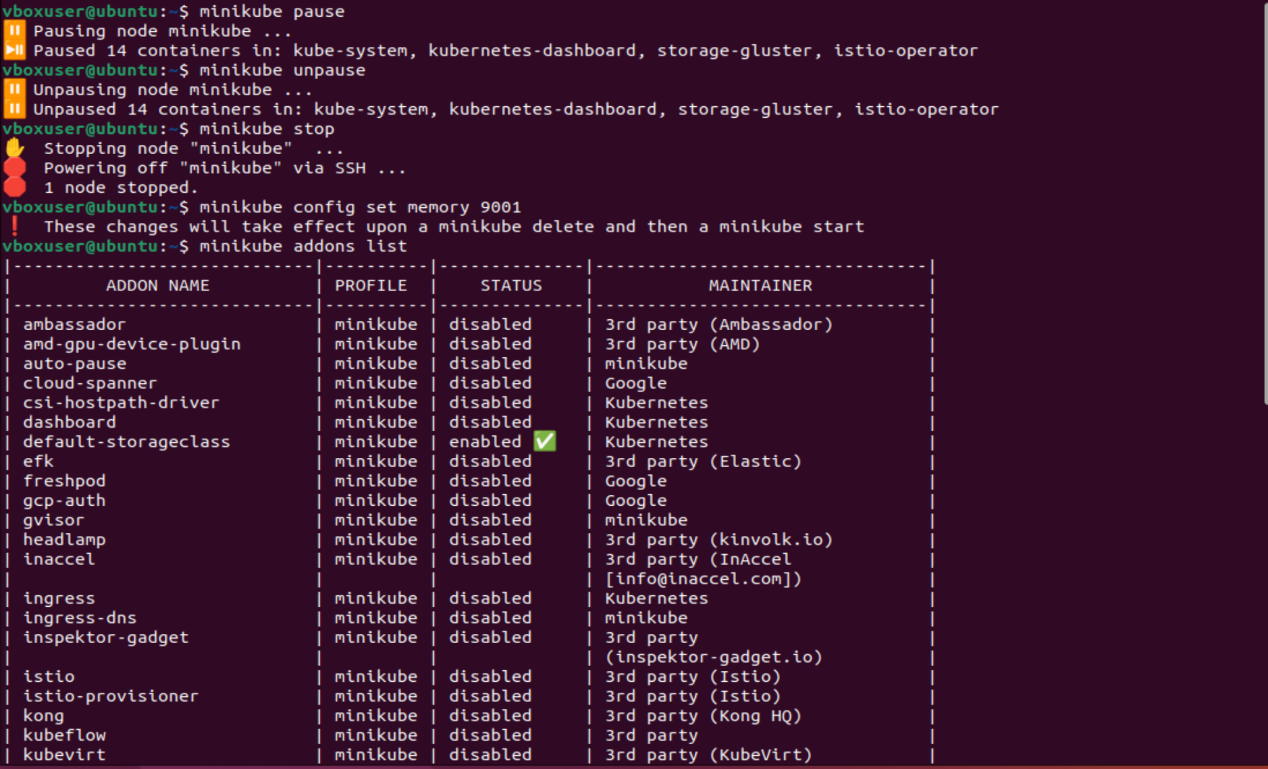
התיאור נוצר באופן אוטומטי

תמונה שמכילה טקסט, גופן, צילום מסך, אלגברה

התיאור נוצר באופן אוטומטי

**שקף 51:**

פקודות לניהול קלאסטר



תמונה שמכילה טקסט, צילום מסך

התיאור נוצר באופן אוטומטי