**תכנון שירותים מבוססי ענן – תרגיל בית 2**

כרים גבארין - 211406343

אחמד גבארין - 314722307

תימאא בסול - 208869222

חלק 1 (יצירת האפליקציה):

ביצענו את השלבים כפי שמתואר בלינק המצורף לחלק הזה. אחר כך, יצרנו את הקובץ Dockerfile שנמצא בתיקייה המצורפת לקובץ הזה. ואז הרצנו את השרת על ידי הפקודות שנמצאות בתום הדף שמצורף לחלק הזה ב- git bash. כדי להתחבר לשרת דרך הדפדפן, הכנסנו localhost:5000 או אפשר גם להכניס 127.0.0.1:5000 שזה אומר שאנחנו מתחברים לפורט מספר 5000 במכונה שמריצה את הקוד. ואז מה שקיבלנו היה דף עם רשימה ריקה, ובמסך של ה- git bash קיבלנו את התגובות של השרת עם קוד 200 שפירושו OK.

חלק 2 (פרישת האפליקציה):

כעת נעלה את ה- docker container שיצרנו ל- GCP Container Registry על ידי תיוג ה- container ואז דחיפתו על ידי שתי הפקודות: docker tag ו- docker push בעזרת git bash, לאחר ביצוע הפקודות האלו, בדקנו את הפרויקט דרך ה- UI של גוגל וקיבלנו את המסך הבא:

![Graphical user interface, application

Description automatically generated]()

כדי ליצור את ה- Managed Instance Group, אנחנו צריכים קודם ליצור VM Template. ביצירת ה- template אנחנו בוחרים בסוג המכונה שתריץ את השרת ובמאפיינים שלה ושל ה- Boot Disk. בנוסף, אנחנו נבחר את ה- image שיצרנו והעלינו ל- GCP Container Registry, רק אחר כך נוכל ליצור את ה- MIG שלנו שיכיל בתוכו לפחות מכונה אחת, ויכיל לכל היותר 10 מכונות באזורים גיאוגרפיים שונים למען אחוז ה- Availability של התוכנית. להלן, צילום מסך ל- UI של ה- Managed Instance Group בפרויקט שלנו עד השלב הזה:

![Graphical user interface, table

Description automatically generated]()

לאחר מכן, יצרנו את מסד הנתונים תחת הלשונית CloudSQL כפי שמתואר בלינק המצורף לחלק הזה.

בקוד של ה- Terraform יצרנו מודולים כדי להפריד בין שתי הסביבות (Dev & Production) והוספנו את קובץ ה- configurations. אחר כך יצרנו מפתח ב- UI של GCP והוספנו את קובץ ה- json לתיקיית הפרויקט, ואז הוספנו את ה- credentials לקוד ה- Terraform כך שימצא את המפתח שיצרנו והוספנו לתיקיית הפרויקט. להלן צילום מסך לקוד יצירת סביבות העבודה ותוצאות שלו:

![Text

Description automatically generated]()

להלן עוד צילום מסך של התמודדות עם סביבות העבודה השונות בפרויקט:

![Text

Description automatically generated]()

לאחר סיום הקוד, הרצנו את הפקודה terraform init שתוריד את כל המשאבים של ה- backend שהקוד צריך, ואז הרצנו terraform plan שתציג את השינויים שיתבצעו על הפרויקט. ובסוף הרצנו את הפקודה terraform apply שביצעה את השינויים שלנו. ואז בדקנו ב- UI של GCP שהפרויקט התעדכן עם המשאבים החדשים.

צילומי המסך הבאים התקבלו מהרצת הפקודה terraform apply עם הקוד שכתבנו על פרויקט ריק:

![Text

Description automatically generated]()

![Text

Description automatically generated]()

![Text

Description automatically generated]()

אחר כך, ביצענו כמה שינויים על הקובץ app.py, ואז כדי לעדכן את הקוד שבענן, נאלצנו למחוק את ה- image הישנה וליצור אחת חדשה, ואז בדומה למה שעשינו בהתחלה, נדחוף את ה- container ל- GCP Container Registry. רק הפעם במקום למחוק את כל המשאבים דרך ה- UI של גוגל שזה לוקח זמן ארוך ואז לבנות את ה- template ואת ה- MIG מהתחלה, את השינויים ייעשו דרך ה- Terraform על ידי הפקודה terraform plan שלוקחת ממש שניות לבצע את כל השינויים על הפרויקט, וזה מה שקיבלנו:

![Text

Description automatically generated]()

בסוף העבודה, וכדי לא לאבד את ה- credits שלנו ב- GCP ביצענו את הפקודה terraform destroy שתמחק את כל המשאבים שיצרנו כולל גם את ה- MIG ואת ה- template כפי שמתואר בצילום הבא:

![Text

Description automatically generated]()