\*מסמך זה בנוסף למסמך הDOCS.

**הפתרון**

אני מבין שאין רצון שאני ארוץ עם כמה סביבות ולכן אנחנו נצא מנקודת הנחה שהסביבה היא סביבת DEV(השינויים העיקריים יהיו בשלב 5, ראה בDOCS)

יצרנו CI CD PIPELINE בGITHUB שבונה קונטיינר לאםליקציית FLASK, מבצע עלייה בדיקות ולאחר מכן מעלה לARTIFACT REPO בGCP.

השתמשתי בkustomize כדי לבנות את הyaml שנדרשים לCD והעלנו בעזרתם לGKE מסוג AUTOPILOT.

לאחר מכן רץ JOB נוסף שבודק שאין שגיאות בDEPLOYMENT(הכוונה בכל הPODים שלו) ומבצע שחזור לגרסה קודמת במידה וקיימים שגיאות(כך מהרגע שהייתה לנו גרסה עובדת אחת אנחנו תמיד נשחזר לגרסה עובדת).

כל התהליך מנוטר בCLEARML והCLUSTER מנוטר בprometheus ומיוחצן לgrafana.

**אתגרים בדרך**

היו כמה אתגרים בדרך, דבר ראשון והכי חשוב היו פה המון כלים שלא השתמשתי בהם לפני כן והיה דיי כיף בתכלס ללמוד לעובד איתם. Gcp, gke, github actions, Prometheus

נתקלתי בבעיות קונפיגורציה שונות עקב כך שבחרתי להשתמש בGKE מסוג AUTOPILOT(אני על משתמש חינמי, רצות לחסוך בעליות) אך זה היה אתגר דיי חביב בסך הכל(במיוחד כשכל הזמן הייתי שנייה מפשוט לעשות קלאסטר רגיל ולקנפג הכל מחדש חחחחחחח, נחמד לראות היה כמה רחוק אני יכול ללכת עם זה)

אתגר נוסף היה לבחור איך אני מבצע כל מיני חלקים במטלה כאשר הייתה לי בחירות מרובות לארכיטקטורה(איך לעשות את הHPA, את הROLLBACK, בדיקות וכו)

ובCLEARML...

**שיפורים לעתיד**

חוץ מהעניין שלא הייתי משתמש בAUTOPILOT מראש(כשכבר הייתי בפנים זה כבר היה אתגר אי אפשר לעזוב P: ) אני אשתמש בחלק הזה במסמך כדי לתאר מה הייתי עושה שונה אם זה היה מרובה סביבות ואמיתי:

1. פיצול לBRANCHים שונים עם PIPELINEים שונים(DEV,PROD וכו. היו לי בראנצים שונים אך הם היו נטו לפיצול על פי הTASKים)
2. העלאה של FLASK לPROD עם ngnix gunicorn
3. הוספה של פיצרים לPIPELINE כמו הרצה גם לא רק כשעושים PUSH(בPR, בלחיצה וכו) ובדיקה על מכונות שונות במידה וצריך לאפליקציה
4. בדיקות יותר עמוקות ושונות(יחידה, אינטגרציה, פונקציונליות וכו)
5. פריסה לתקיות בGIT(שמתי הכל במקום אחד כי לא רואה צורך באינספור תקיות שיש בהן משהו אחד לאפליקציה כל כך פשוטה זה מיותר, רציתי לשמור על פשטות)
6. AUTOSCALING עם שימושים אמיתיים
7. פריסה חכמה לPROD(blue green וקדומה)
8. ניטור עמוק וחכם בGRAFANA שכולל גם ALERTים(בתכלס נראה שהmonitoring של GCP דיי טוב, יש מצב הייתי משתמש בו במקרה אמיתי. הוא תומך גם בpromql)
9. Rollbackים עמוקים שמסתמכים גם על הALERTים מהסעיף הקודם
10. לא הייתי משתמש בclearml לנטר את הPIPELINE...