בתרגיל יצרנו שני סוגי רכיבים:

WORKER:

* סקריפט שרץ אקטיבית, ודוגם את הEndpoint שהרים אותו, לראות אם יש משימות לעשות.
* לאחר שדגם 3 פעמים ולא קיבל תוצאות יהרוג הWORKER את עצמו ואת הEC2 עליו הוא יושב ויעדכן בכך את הEndpoint שהרים אותו.
* את העבודה שעשה יישלח לIP שקיבל בתור פרמטר של העבודה. הכתובת הזו אמורה להפנות לEndpoint שממנו הלקוחות יבקשו את התשובות לעבודות.

Endpoint:

* ישנם שני instanceים של הרכיב. מבחינת קוד שניהם יהיו זהים אך הלקוחות שפונים ייפנו לאחד מהם על מנת להקצות עבודות ולשני בכדי לקבל את התוצאה.
* את הבקשות הרכיב שם בתור פנימי
* כאשר מצטברות הודעות בתור הפנימי ירים הרכיב Workerים שייפנו אליו ויפרקו את התור עד מספר תקרה שהגדרנו.
* בהודעות שבתור אנחנו נשמור את הIP של רכיב הEndpoint השני שאליו ייפנו על מנת לקבל את תוצאות העבודה.
* כאשר Worker פונה כדי לקבל הודעה שבתור ואין הודעות, נשתמש בפיצ'ר שגורם לבקשה להמתין 10 שניות כדי לראות אם בכל זאת תגיע הודעה.
* כאשר Worker הורג את עצמו הוא ייפנה לAPI שמעדכן שהוא לא רלוונטי יותר ובכך נדע שיש לנו Worker אחד פחות ונוכל להוסיף עוד בלי להגיע לתקרת הWorkerים שהגדרנו.
* לרכיב גם מונגש ממשק swagger שמאפשר לנו ידנית לבדוק את כל פעולות הAPI בכתובת: http://my\_endpoint\_ip:5000/apidocs.

מקרי כשל אפשריים:

* Endpoint נופל או מתנתק מהרשת – הרכיב הזה מחזיק בסטייט של העבודות שיש לבצע ואלו שבוצעו. לכן כדי שנוכל להיות עמידים לנפילות שלו נרצה שגם הוא יהיה סקיילאבילי וחסר סטייט. כלומר, להחזיר את תורי העבודות בתור או cache חיצוני כגון רדיס או קפקא. כך שנוכל להרים מספר גדול של Endpointים ואם אחד או חלק ייפלו נוכל להמשיך לעבוד ללא הפרעות.
* Worker נופל או מתנתק מהרשת – נוכל לבדוק בEndpoint כמה זמן עבר מאז כל נסיון קריאה מתור העבודות של כל Worker ואם עבר יותר מדי זמן נשלח פקודת הריגה (שאולי גם תכשל) ונוריד אותו ממספר הWorkerים החיים שיש לנו

נשים לב ש:

* הרמת הEndpoints לוקחת לא מעט זמן
* הרמת כל Worker לוקחת גם לא מעט זמן, כך שגם יכולים להיות מצבים שהתור יהיה מלא ולא יגיעו תשובות לEndpoint שמחזיר תשובות למשך מספר דקות. בנוסף אם יהיה עומס בקשות יכול להיות שיועלו מספר Workerים לפני שייפרקו הודעות מהתור

תודה!