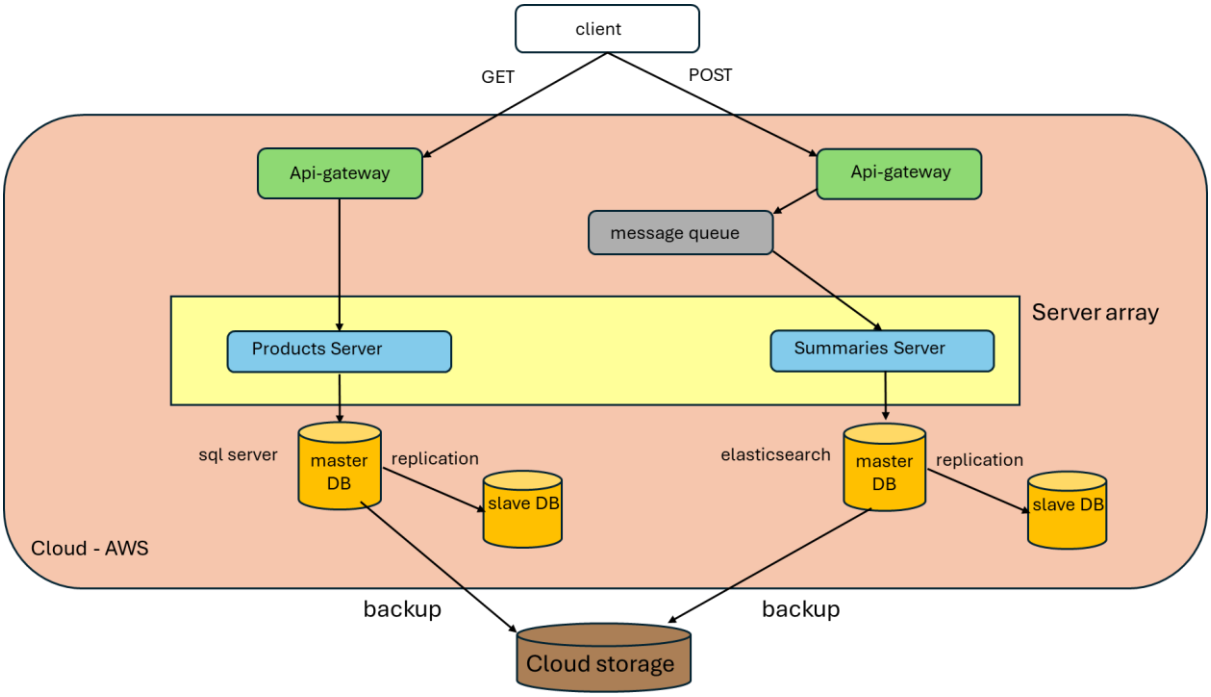
1. בחרתי בAWS מכיוון שהפלטפורמה שלהם וותיקה ומיקפה יותר, מה שמאפשר מגוון גדול יותר של שירותים למערכת.
החלטתי להשתמש ברכיבים הבאים:
1. Api-gateway – על מנת לנתב את הבקשות מהלקוח למופעים השונים של כל אחד מהשרתים. בנוסף מאפשר יכולות אבטחה וניטור. בנוסף, לgateway של ה-productserver נוכל להוסיף גם cach כך שנוכל למעט בפנייות לשרת.
2. rightScale – מאפשר horizontal scaling בצורה אוטומטית לשרתים שלנו.
3. Storage backup – גיבוי לDB שלנו.

המערכת בנויה משני שירותי backend עיקריים, אשר מספקים את השירותים הנחוצים ללקוח.

a. משיכת מידע על המוצרים בחנות:
   i. נשלחת בקשת GET מהלקוח לload balancer שיושב בענן.
   ii. הLB מנתב את הבקשה לאחד ממופעי הproductServer.
   iii. הproductServer מושך את המידע מהDB (sql).
   iv. המידע מוחזר ללקוח.
b. שמירת סיכום הזמנה:
   i. הלקוח שולח בקשת POST עם פרטי הזמנה ל gateway.
   ii. הgateway שומר את פרטי ההזמנה בmassage queue, ומחזיר תשובה ללקוח שההזמנה נשמרה.
   iii. הMQ מעביר את פרטי ההזמנה לאחד ממופעי הsummariesServer שפנוי.
   iv. הsummariesServer שומר את המידע על ההזמנה בDB (elasticsearch).

2.



.3

a. Api-gateway – מהווה את שער הכניסה של הבקשות מצד השרת למערכת שלנו. מאפשר לנתב את הבקשות מהלקוח למופעים השונים של כל אחד מהשרתים (load balancing) ומונע SPOF (single point of failure). בנוסף מאפשר יכולות אבטחה וניטור. בנוסף, לgateway של ה-productserver נוכל להוסיף גם cache כך שנוכל למעט בפניות לשרת.
b. Message queue – מאפשר לתהליך הכתיבה של סיכומי הקניה לשרת להתבצע בצורה אסינכרונית, אמינה (ללא חשש של איבוד הזמנה בשל אי נפילה של שרת) וללא עיקוב של הלקוח.
c. rightScale – מאפשר horizontal scaling בצורה אוטומטית לשרתים שלנו. מופיע כserver array.
d. Master-slave db – מונע SPOF ומאפשר קונסיסטנטיות של המידע. בנוסף מאפשר ביצועים יותר טובים מכיוון שניתן לקרוא מכל השרתים.
e. Cloud storage – לגיבוי, מאפשר שרידות של מהערכת.