digital whisper
Technologic Papers

איך להעביר מכתב אהבה

מאת אלכסיי זלמנוביץ' וניצן שולמן

הקדמה

ריצת מרתון היא ריצה למרחק של 42.195 קילומטרים. מקור שמה של הריצה הוא באגדה עממית אודות פידיפידס, שליח אתונאי אשר רץ מהעיר מרתון ועד אתונה בשנת 490 לפנה"ס כדי להודיע על הניצחון היווני נגד הפולש הפרסי בקרב בעיר. באגדה מסופר שלאחר שהודיע את ההודעה השליח מת.

בימינו קצת יותר קל לתקשר ולהעביר הודעות. אפשר לתקשר בעזרת מילים שיוצאות מהפה ואפשר גם בקריאה של מילים (כמו המאמר הזה). בנוסף ניתן לתקשר גם בלי מילים! לדוגמא בעזרת אותות מורס או שפת הסימנים.

בעולם המחשבים כמו בעולם של בני האדם, תקשורת היא מרכיב ראשי. בספר "Ender's Game" (המשחק של אנדר) נאמר:

"So the whole war is because we can't talk to each other."

בין אם זה בין מחשבים שונים או תוכנות שונות, איך שהתקשורת תתנהל תשפיע על המערכת עצמה ועל השימוש בה. בתכנות אחד הדברים הנפוצים והחשובים הוא להפריד חלקים גדולים של הקוד לכמה חלקים קטנים ולקרוא להם microservices; מערכות (services) קטנות (micro) שביחד מאפשרות התנהלות כמערכת אחת גדולה. למשל מערכת אשראי דמיונית, שמטרתה לחשב את הסכום שעל המשתמש לשלם ולגבות את הכסף מהכרטיס עצמו.

בהתחלה החישוב של הסכום פשוט, לפי המחיר של המוצר פלוס מע"מ. עם הזמן החישוב הופך למסובך יותר ויותר. דברים כמו הצורך להוסיף התחשבות בעיר ובמע"מ של אותה עיר.

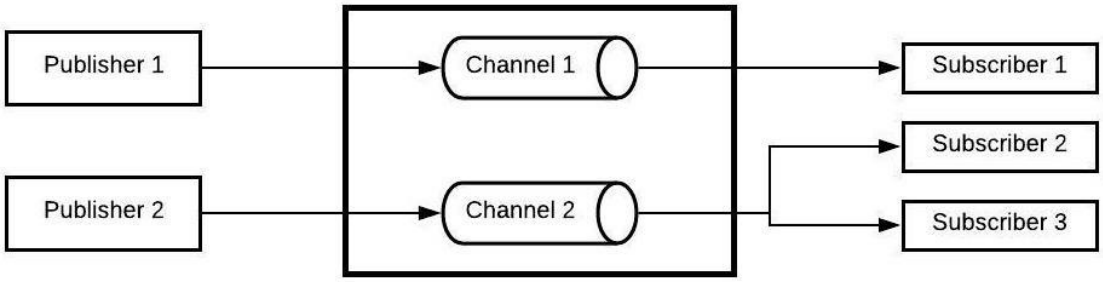
במקביל, חברות האשראי רק הופכות את התשלום ליותר ארוך. הן בודקות את המיקום שממנו משלמים ומתחשבות בהיסטורית קניות של המשתמש של הכרטיס. הכל בשביל לאמת את התשלום עצמו.

הצגת מימושים

Redis

Redis או REmote DIctionary Server - הוא שרת שאפשר לאחסן בו כל מיני מבני מידע קלאסיים, מחרוזות, רשימות, מילונים ועוד. הפרויקט הוא open source והתחיל באזור 2009. מתחילתו הוא פרח וכבר יש תמיכה בו בכל שפות התכנות הגדולות.

מנגנון התקשורת שקיים ב-Redis נקרא publish ו-subscribe או בקיצור **pubsub**. במנגנון הזה ניתן להגדיר channel (נושא) מסוים שאפשר לפרסם ולהאזין להודעות ממנו. לדוגמא אם נרצה ליצור חדר צ'אט עבור משחק מחשב נוכל להשתמש ב-Redis, ברגע שמישהו מצטרף לחדר משחק הוא עושה subscribe ל-channel של החדר ומתחיל לקבל ולפרסם הודעות ממנו.



Redis Pub/Sub

בתמונה אפשר לראות שיש שתי ישויות שונות שכל אחת מהן מפרסמת הודעות עבור נושא מסוים. ל-channel הראשון יש ישות אחת שמאזינה ול-channel השני יש שתיים, כל הודעה המתפרסמת ב-channel מסוים תגיע לכל המאזינים שלו.

אחד הפיצ'רים של מנגנון ה-pubsub הסטנדרטי הוא שהודעות לא נשמרות! אני מסכים, זה נשמע מוזר אבל בואו נחזור לחדר צ'אט במשחק מחשב. בתור שחקן כנראה שלא יעניין אותי הודעות שנרשמו לפני שהגעתי, וגם אין טעם בשמירת ההודעות שהיו בחדר לאחר שהמשחק נגמר.

דוגמת שימוש

שימוש של Redis יכול להיות עבור אפליקציה שמעדכנת את המיקום של כל מיני אנשים. במידה ואני באפליקציה ורוצה לשתף את המיקום של אני אפרסם את המיקום שלי תחת channel מסוים, לדוגמא תחת השם שלי. כל מי שירצה לדעת איפה אני נמצא יכול להאזין ל-channel ולקבל מידע על איפה אני נמצא. בכיוון ההפוך, אני אאזין לכל האנשים שאני רוצה לקבל מידע על איפה הם נמצאים. הפיצ'ר שהודעות לא נשמרות ב-Redis מתאים לאפליקציה הזאת, מיקום לא עדכני לעכשיו לא יהיה מעניין עבורי.

RabbitMQ

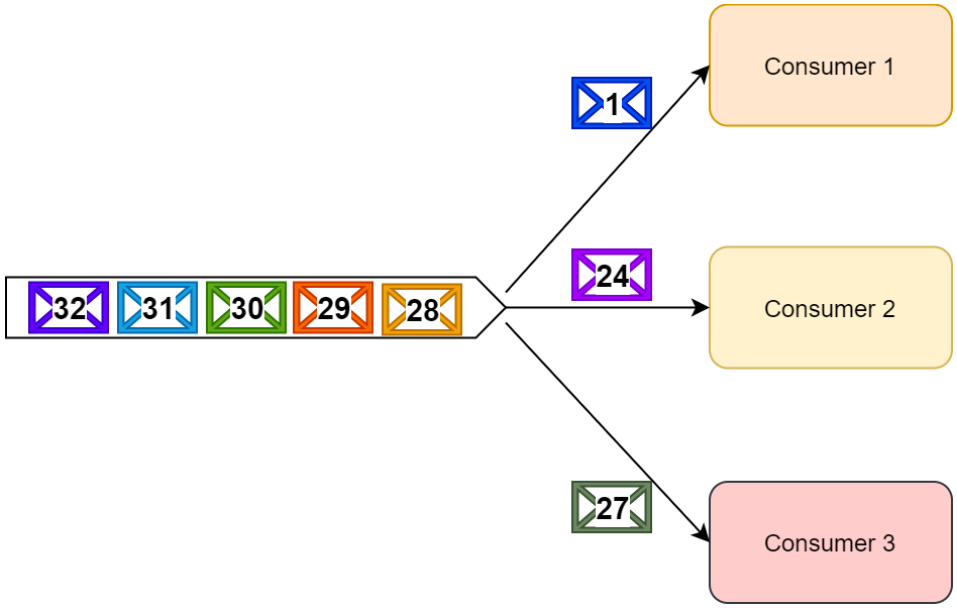
RabbitMQ היא מערכת open source שמוגדרת כ-message-broker; מתווך הודעות.

המבנה הבסיסי של RabbitMQ בנוי משלושה חלקים:

1. Queue - מבנה נתונים שמאחסן מידע בצורה של הראשון שנכנס הוא הראשון שיוצא (נקרא גם FIFO).
2. Producer - גורם ששולח מידע אל ה-queue.
3. Consumer - גורם שמתחבר ל-queue מסוים ומקבל את ההודעות ממנו.

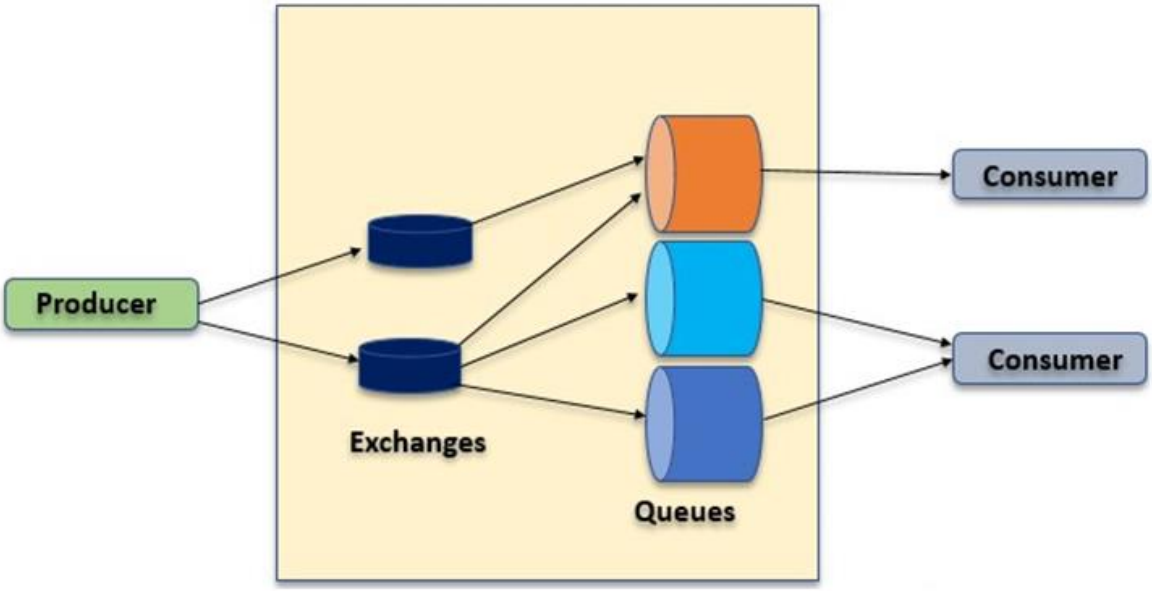
כאשר consumer קולט הודעה, הוא מוציא אותה מה-queue ומעבד אותה. אחרי שהוא מסיים, הוא עובר להודעה הבאה שמחכה.

ישנה את האפשרות של לחבר מספר של consumers לאותו queue, ובכך "לייעל" את קריאת ההודעות שנשלחות ל-queue:



כפי שאפשר לראות, consumer 1 תקוע על הודעה 1 בזמן ששאר ה-consumers מעבדים את ההודעות בקצב שלהם ומוציאים מה-queue.

על גבי המבנה הזה ישנו גם את ה-exchange, שאחראי על ההעברה של המידע בין ה-producer ל-queues הרלוונטיים:



[מקור: educba.com]

Exchanges הם ה"מוח", והם מחוברים ל-queues. אפשר להגדיר שההודעות יעברו בצורות מסוימות ל-queues שונים. לדוגמא fanout - שפשוט מעביר כל הודעה שהוא מקבל לכל ה-queues שמחוברים אליו.

דוגמת שימוש

RabbitMQ משומש המון כמערכת לשליחת "משימות" שיש לבצע. היא טובה בשביל זה כי אפשר להגדיר queue של משימות, ולהגדיר "מבצעים" שמתחברים ל-queue ומבצעים אותן. בגלל המבנה של RabbitMQ, אין תלות בינם לבין עצמם ולכן ברגע שמבצע אחד תקוע על משימה כלשהי, השאר עוברים למשימות הבאות שיש לבצע בלי לחכות.

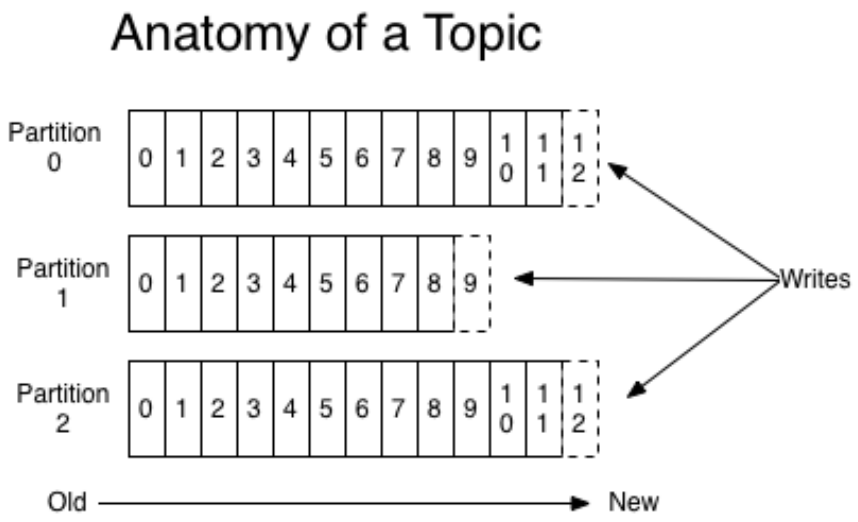
דוגמא חיה היא מערכת חיוב אשראי. אפשר להגדיר queue של חיובים לביצוע ו-consumerים שמקבלים את המשימות ומחייבות את הכרטיס. ה-producer שולח כל פעם שמעבירים כרטיס את "משימת החיוב" שיש לבצע.

Kafka

בשונה מהדוגמאות האחרות שהבאנו עד כה, Kafka מוגדר כ-Messaging System (מערכת להעברת הודעות). הוא פותח במקור על ידי LinkedIn וב-2011 נהפך ל-open source כאשר Apache כרגע המתחזקים שלו.

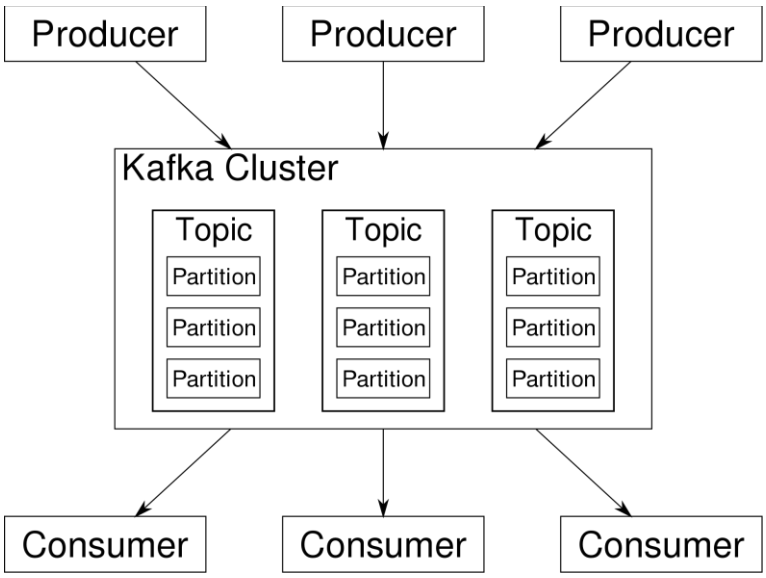
ב-Kafka המידע מחולק על פי **topics**(נושאים). כל נושא בנוי ממספר מוגדר של **partitions**כאשר בתוכם המידע של הנושא הולך לשהות.

ה-partitions בנויים כמו שרשרת בלתי נגמרת של מידע, כאשר כל הודעה חדשה נוספת לסוף של אחד ה-partitions.

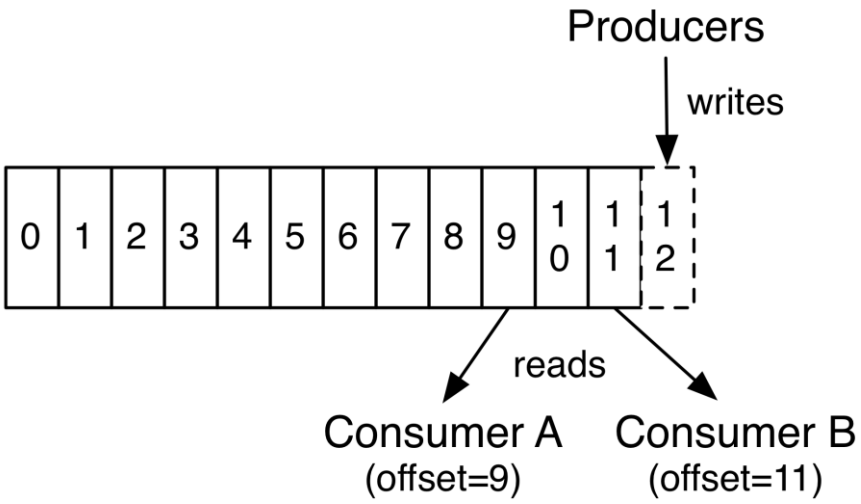


יש לציין כי המידע שמור על הדיסק ויעלם רק אחרי כמות מוגדרת של זמן.

אל ה-topic מתחברים producers ו-consumers:



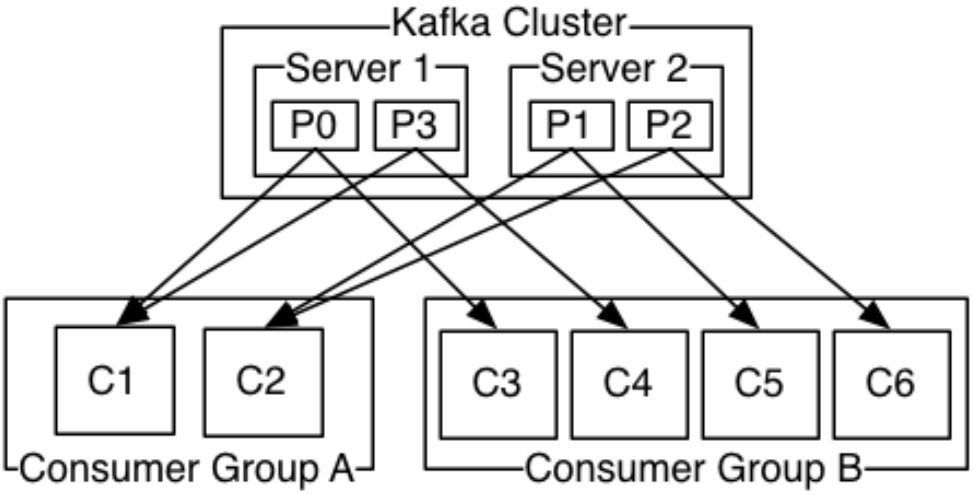
כאשר producer שולח מידע ל-topic מסוים, המידע מגיע ל-partition אקראי כלשהו. Consumer מהצד השני מתחבר לנושא מסוים ויכול לקרוא מכל ה-partitions מאיפה שהוא רוצה: בין אם זה מההודעה הכי ישנה שנמצאת ב-topic לבין לקרוא רק הודעות חדשות מאותו רגע של התחברות.



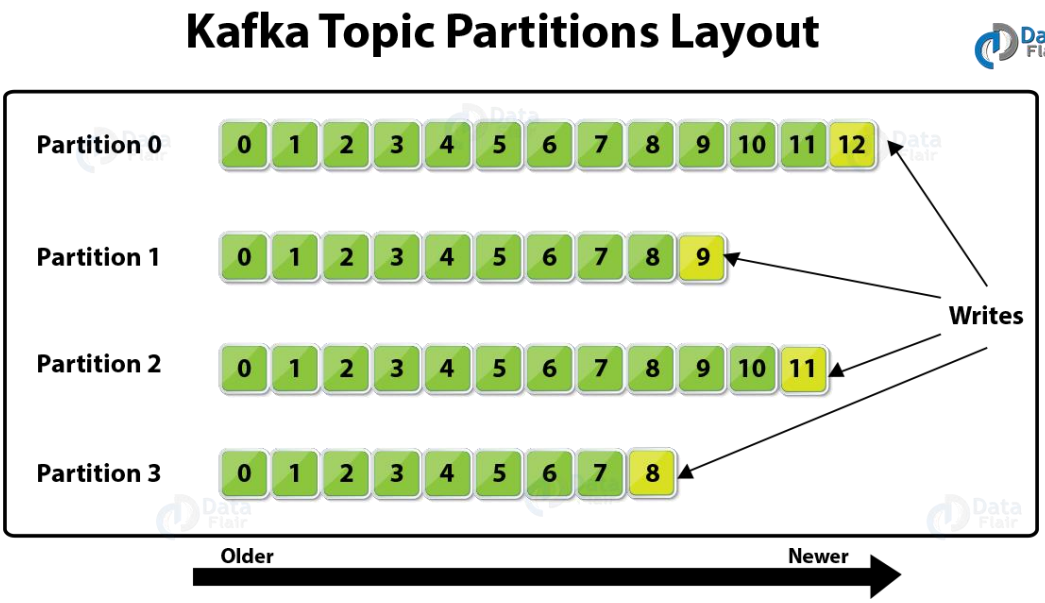
פיצ'רים נוספים

למבנה הזה Kafka מוסיף עוד פיצ'רים מדהימים; כאשר העיקריים הם:

1. Consumer Groups - מאפשר להגדיר קבוצה של consumers שמחלקים את ה-partitions של אותו topic שווה ביניהם, וכך בעצם מקבלים הודעות כקבוצה. זאת אומרת שאם יש לי 3 partitions ו-3 consumers המוגדרים תחת אותה קבוצה, כל consumer יקבל הודעות מ-partition אחר. זה מאפשר לנו "קריאה במקביל" של מידע.



2. Message ID - אפשר לסמוך על זה שסדר ההודעות ב-partition הוא כרונולוגי (כי הוא מתווסף לסוף). עם זאת, המידע בין partitions שונים הוא לא בהכרח כרונולוגי.



הודעה 11 ב-partition 2 לא בהכרח יותר חדשה מהודעה 8 ב-partition 3. אז מה עושים אם רוצים לשלוח רצף של הודעות שבהכרח יתקבלו באותו רצף השליחה? Message ID!

יש לנו את האפשרות לתת ID להודעות. ה-ID יגרום להודעות להיכנס לאותו **partition** ובכך ליצור את הרצף שאנחנו רוצים. זאת אומרת שאם היינו רוצים לשלוח ל-topic 3 הודעות שבהכרח ייקראו אחת אחרי השניה, אפשר להוסיף להם ID=1 ואז כל הודעה עם ID=1 תישלח ל-partition 0 (ה-partition אקראי, אבל ההודעות ייכנסו רק אליו).

דוגמת שימוש

דוגמא נפוצה היא log aggregation; שזה בעצם לקחת logs מהרבה מקומות ולשים אותם במקום משותף אחד. נגיד וישנה מערכת שבה יש לנו עשרות מערכות שונות, שבכל אחת מהן logs משל עצמה. בשביל לעקוב בצורה נוחה אפשר להשתמש ב-kafka בשביל לשלוח את ה-logs ל-topics ייעודיים ובכך לאגד את ה-logs.

סיכום

במאמר הוצגו שלושה מימושים של העברת הודעות השונים במהותם אחד מהשני. חשוב לדעת שגם אפשר לקנפג כל מימוש כך שיתנהג יותר כמו האחר. אפשר לגרום ל-redis להתנהל יותר דומה ל-RabbitMQ וגם הפוך. העובדה שכל המימושים האלה הם open source מקלה על הקינפוג והשינוי שלהם שיתאימו לשימושים שאנו צריכים.

בסופו של דבר מה שחשוב הוא הבסיס של כל מימוש. Redis מתנהל כמערכת הודעות שיותר מותאמת ל-"fire and forget". לעומת זאת, RabbitMQ מאוד טוב עבור שליחת משימות. בשונה מהם, Kafka מאפשר קליטת הודעות בצורה רצופה ומקבילית, הבנויה עבור רצף של מידע כמו logs ומיקום.

הצגנו במאמר זה רק את קצה הקרחון של העולם התקשורתי בין רכיבים במערכות גדולות, כדי להראות את השימושים השונים והמימושים הנמצאים בו. בסופו של דבר יש עוד מלא סוגי תקשורת בעולם, אנחנו בתור המפתחים צריכים למצוא את הדבר המתאים למקרה הספציפי שנתקל בו. האם בכלל המערכת צריכה תקשורת ברשת בינה לבין עצמה? איך היא תתנהל? מה נעשה עם המידע? איך הוא ייראה? האם עליו להיות פרסיסטנטי על הזיכרון או להיעלם אם אף אחד לא מוכן לקבל אותו כרגע? כל אלה הן שאלות שצריכות להיות לנו בראש כשאנחנו מעצבים את המערכות של המחר.

על הכותבים

אלכסיי זלמנוביץ' - אוהב לעצב מערכות וקפה עם הל

https://www.linkedin.com/in/alexey-zelmanovich

ניצן שולמן - אוהב חמאת בוטנים וביקורת טובה, מוזמנים לפנות במייל:

nitzanshulman@gmail.com

ביבליוגרפיה

Redis

- Redis Official Site
- What to Choose for Your Synchronous and Asynchronous Communication Needs-Redis Streams, Redis Pub/Sub, Kafka, etc.
- https://realpython.com/python-redis/
- https://en.wikipedia.org/wiki/Redis

RabbitMQ

- RabbitMQ Official Site

Kafka

- Kafka Official Site
- Apache Kafka Series - Learn Apache Kafka for Beginners v2
- A Brief History of Kafka, LinkedIn's Messaging Platform

Comparisons

- Kafka vs RabbitMQ part 1
- Kafka vs RabbitMQ part 2
- Difference Between RabbitMQ vs Kafka
- Kafka vs. Redis: Log Aggregation Capabilities and Performance
- Kafka vs Redis discussion
- Difference Between RabbitMQ vs Redis