การปรับใช้ ซอฟต์แวร์

กระบวนการ

รัศมีทิพย์ วิตา

7 กันยายน 2023

เกี่ยวกับฉัน

- รัศมีทิพย์ วิตา

AWS Academy Graduate - AWS Academy Machine Learning Foundations
Amazon Web Services Training and Certification

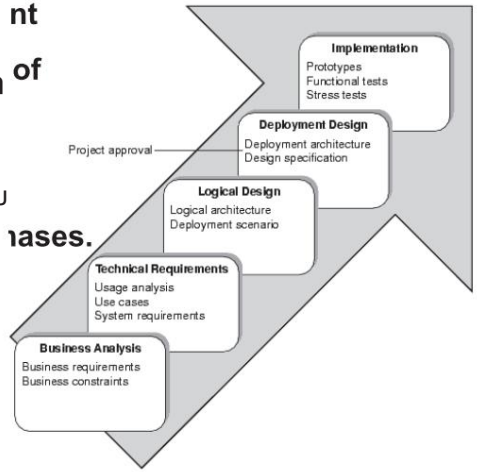
AWS Academy Educator
Amazon Web Services Training and Certification

AWS Academy Graduate - AWS Academy Introduction to Cloud Semester 1
Amazon Web Services Training and Certification

AWS Academy Graduate - AWS Academy Cloud Foundations
Amazon Web Services Training and Certification

กระบวนการวางแผนการปรับใช้

- การใช้งานสำเร็จ nt
  การวางแผนเป็นผลจาก of
  การเตรียมการอย่างระมัดระวัง
  การวิเคราะห์และการออกแบบ
  ผ่านชุดของขั้นตอน



https://docs.oracle.com/cd/E19199-01/817-5759/intro.html

ขั้นตอนการออกแบบการปรับใช้งาน

- ออกแบบสถาปัตยกรรมการปรับใช้
  - การทำแผนที่สถานการณ์การปรับใช้งานกับ
    สภาพแวดล้อมทางกายภาพ
  - ฮาร์ดแวร์จริงที่จำเป็นต่อ การตอบสนองของระบบ
    ข้อกำหนด และการกำหนดกลยุทธ์ในการ
    ปรับสถาปัตยกรรมการใช้งานให้เหมาะสมที่สุดเพื่อตอบสนอง
    การพิจารณางบประมาณ

ขั้นตอนการออกแบบการปรับใช้งาน

- การออกแบบสถาปัตยกรรมการปรับใช้ส่งผลโดยตรง ประสิทธิภาพของแอปพลิเคชัน ความน่าเชื่อถือ และ ความสามารถในการขยายขนาด
- ต้องมีการวางแผนและการพิจารณาอย่างรอบคอบ ความ ต้องการและเป้าหมายเฉพาะ ของการสมัคร และผู้ใช้

สถาปัตยกรรมการปรับใช้

- เซิร์ฟเวอร์และโครงสร้างพื้นฐาน
  - เซิร์ฟเวอร์จริงหรือเสมือน อุปกรณ์เครือข่าย โซลูชั่นการจัดเก็บข้อมูล
  - โครงสร้างพื้นฐานบนคลาวด์
- ส่วนประกอบแอปพลิเคชัน
  - เว็บเซิร์ฟเวอร์ แอพพลิเคชั่นเซิร์ฟเวอร์ ฐานข้อมูล โหลดบาลานเซอร์ และไมโครเซอร์วิส

สถาปัตยกรรมการปรับใช้ [2]

- ความสามารถในการขยายขนาดและการปรับสมดุลโหลด
  - วิธีที่แอปพลิเคชันสามารถปรับขนาดเพื่อจัดการได้ ระดับการจราจรหรือโหลดที่แตกต่างกัน
- การจัดเก็บข้อมูลและฐานข้อมูล
  - การตัดสินใจเกี่ยวกับการจัดเก็บข้อมูล ได้แก่ ฐานข้อมูล (SQL, NoSQL), การแคช กลไกและการจัดเก็บไฟล์

สถาปัตยกรรมการปรับใช้ (3)

- ความปลอดภัย
  - มาตรการรักษาความปลอดภัย เช่น ไฟร์วอลล์ การบุกรุก ระบบการตรวจจับ การเข้ารหัส และการเข้าถึง ควบคุม
- เครือข่าย
  - การกำหนดการตั้งค่าเครือข่ายได้แก่ กฎการกำหนดเส้นทาง ซับเน็ต และไฟร์วอลล์

สถาปัตยกรรมการปรับใช้ (4)

- โมเดลการใช้งาน
  - สถาปัตยกรรมการปรับใช้สามารถจำแนกได้เป็นรุ่นต่างๆ ขึ้นอยู่กับข้อกำหนด
    เช่น แบบเสาหิน ไมโครเซอร์วิส แบบไร้เซิร์ฟเวอร์ และแบบอิงคอนเทนเนอร์
- การกู้คืนความเสียหายและความพร้อมใช้งานสูง
  - แผนสำหรับการกู้คืนระบบและความพร้อมใช้งานสูงควรเป็นส่วนหนึ่งของสถาปัตยกรรมการใช้งาน
    รับประกันเวลาหยุดทำงานและการสูญเสียข้อมูลน้อยที่สุดในกรณีดังกล่าว
    ความล้มเหลว

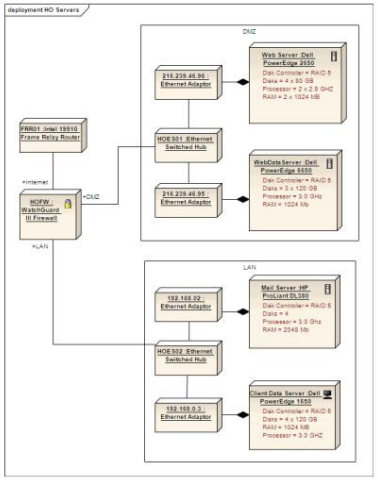
สถาปัตยกรรมการปรับใช้ (5)

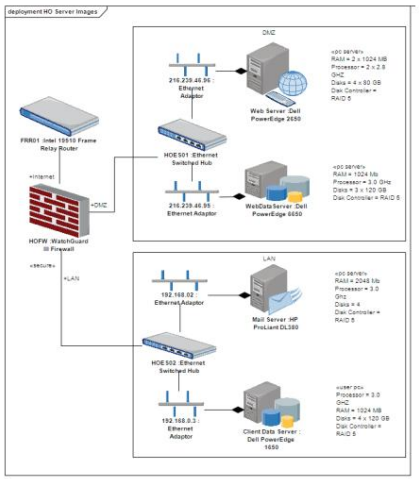
- การตรวจสอบและการบันทึก
  - การใช้ เครื่องมือตรวจสอบ และการบันทึก
    กลไกช่วยในการติดตามแอปพลิเคชัน
    ประสิทธิภาพ การระบุปัญหา และการแก้ไขปัญหา
- การบำรุงรักษาและการอัพเดต
  - กลยุทธ์สำหรับการอัปเดตแอปพลิเคชัน, แพตช์
    การจัดการและการควบคุมเวอร์ชันถือเป็นสิ่งสำคัญ
    ทำให้ระบบที่ปรับใช้ทันสมัยและปลอดภัยอยู่เสมอ

แผนภาพการปรับใช้

- แผนภาพการใช้งานเป็นแผนภาพที่แสดง
  การปรับใช้ทางกายภาพของส่วนประกอบของระบบ
- มันแสดงให้เห็นว่าระบบจะถูกนำไปใช้งานอย่างไรและที่ไหน
- สิทธิประโยชน์
  - เห็นภาพสถาปัตยกรรมทางกายภาพของระบบ
  - ระบุปัญหาคอขวดของประสิทธิภาพที่อาจเกิดขึ้นหรืออื่นๆ
    ปัญหา.
  - ใช้ในการวางแผนและปรับใช้ระบบซอฟต์แวร์
  - ใช้เพื่อจัดทำเอกสารการใช้งานระบบ

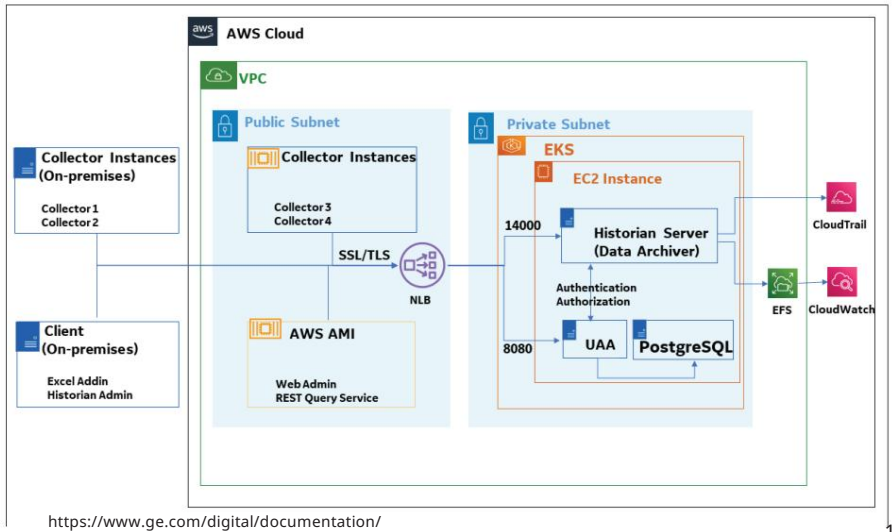
แผนภาพสถาปัตยกรรมการปรับใช้ -
บนสมมติฐาน





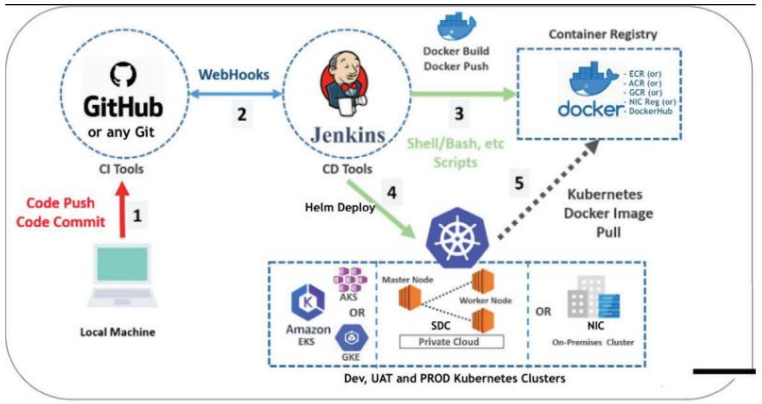
https://sparxsystems.com/enterprise_architect_user_guide/16.1/modeling_lanวัด/deploymentdiagram.html

แผนภาพสถาปัตยกรรมการปรับใช้-บนคลาวด์



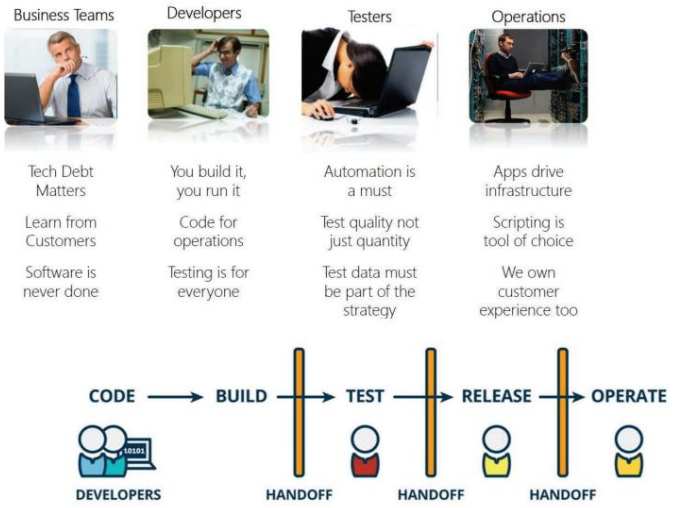
https://www.ge.com/digital/documentation/cloud- ประวัติศาสตร์/version2022.1/r_deployment_architecture.html

แผนภาพสถาปัตยกรรมการปรับใช้ - คอนเทนเนอร์



https://core.digit.org/platform/architecture/deployment-architecture

วงจรชีวิตการพัฒนาซอฟต์แวร์ - SDLC



เป้าหมายและกลยุทธ์การปรับใช้

https://amazic.com/choosing-the-right-application-deployment-strategy/

เป้าหมายและแนวปฏิบัติ

- เมื่อเวลาทำงานเป็นสิ่งสำคัญ ให้คิดถึงการ ลดเวลาหยุดทำงานให้เหลือน้อยที่สุด
- คิดถึง การย้อนกลับ ในกรณีที่เกิดข้อผิดพลาดระหว่างการใช้งานหรือระหว่างนั้น
  การทดสอบ (ความเสถียรหรือประสิทธิภาพ)
- ใส่ทุกอย่าง (สคริปต์และการกำหนดค่าทั้งหมด) ไว้ในการควบคุมเวอร์ชัน เพื่อให้คุณสามารถใช้ไปป์ไลน์ CI/CD เพื่อ
  ปรับใช้แอปพลิเคชันเวอร์ชันใหม่ของคุณได้อย่างรวดเร็ว และคุณสามารถติดตามและติดตามได้
- การปรับใช้ควรสอดคล้องกันและทำซ้ำในสภาพแวดล้อมต่างๆ เพื่อให้การส่งเสริมแอปพลิเคชันจาก
  สภาพแวดล้อม DEV ไปจนถึง TEST ไปจนถึง ACC ไปจนถึง PROD ได้อย่างราบรื่น
- คุณเคยคิดถึง ความเข้ากันได้แบบย้อนหลังหรือไม่? ตัวอย่างเช่น:
  ให้บริการฐานข้อมูลเดียวที่มีการเชื่อมต่อแอปพลิเคชันหลายเวอร์ชัน
- ดำเนินการวิเคราะห์สาเหตุที่แท้จริง (RCA) ในกรณีที่มี การใช้งาน
  ความล้มเหลว.

กลยุทธ์การปรับใช้แอปพลิเคชัน

กลยุทธ์	ขั้นตอน	ข้อได้เปรียบด้านเทคนิค	ธุรกิจ ข้อได้เปรียบ	ข้อเสีย
การอัปเกรดแบบ แทนที่แทนที่เวอร์ชัน X ด้วยเวอร์ชัน Y		เรียบง่าย คุ้มราคา- มีประสิทธิภาพ	คุ้มค่าสำหรับ แอปพลิเคชันรุ่นเก่า	หยุดทำงาน, ความสามารถในการปรับขนาดที่จำกัด
ฟ้าเขียว การปรับใช้	ปรับใช้ใหม่ เวอร์ชันเคียงข้างกัน เวอร์ชันก่อนหน้า	การทดสอบแบบขนาน น้อยที่สุด หยุดทำงาน	ไม่ต้องหยุดทำงาน ต่อเนื่อง การดำเนินการ	ราคาแพง เวลา- การบริโภค
การอัปเดตแบบกลิ้ง	แทนที่อินสแตนซ์ ทีละคน	ไม่ต้องหยุดทำงาน ความเท่าเทียม	ไม่ต้องหยุดทำงาน ลดต้นทุน	ขาดการจราจร การควบคุม, การจัดการ API หลายรายการ
การทดสอบ A/B	เสิร์ฟที่แตกต่าง เวอร์ชันที่จะทดสอบ	ควบคุมเต็มรูปแบบ ง่าย ย้อนกลับ	รวดเร็ว การทดลอง ลูกค้า ข้อมูลเชิงลึกด้านพฤติกรรม	
คานารี่ การปรับใช้	ค่อยๆ เพิ่มขึ้น สัญจรไปมาใหม่ เวอร์ชัน	การทดสอบแบบค่อยเป็นค่อยไป การพลิกกลับอย่างรวดเร็ว	คุณสมบัติที่รวดเร็ว การทดสอบคำติชม ของสะสม	ไม่ล้มเหลว-Safe
เงา การปรับใช้	ทดสอบการผลิต โหลดใหม่ครับ คุณสมบัติ	ไม่มีโหลดเฉพาะ สภาพแวดล้อมการทดสอบ เลียนแบบจริง การจราจร	โหลดที่สมจริง การทดสอบขนาดเล็ก เพิ่มขึ้นล้มเหลว เร็ว	การจัดการข้อมูล ความท้าทาย

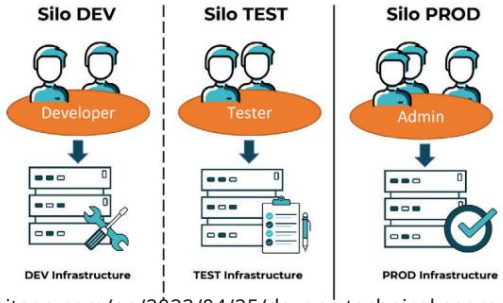
แนวคิด DevOps

ไซโลไอที

- นักพัฒนาและผู้ดูแลระบบพูดไม่เหมือนกัน

  ภาษา. งานที่แตกต่างกันมีการจัดลำดับความสำคัญที่แตกต่างกันและ

  มักมีวัฒนธรรมการทำงานและวิธีการทำงานที่แตกต่างกัน



https://www.amitego.com/en/2022/04/25/devops-technical-segregation-of-duties/

DevOps คืออะไร

- DevOps คือชุดแนวทางปฏิบัติที่ผสมผสานกัน

  การพัฒนา ซอฟต์แวร์ (Dev) และ การดำเนินงาน ด้านไอที

  (ปฏิบัติการ).

- มีเป้าหมายเพื่อลดอายุการพัฒนาระบบให้สั้นลง

  วงจรและให้การส่งมอบอย่างต่อเนื่องสูง

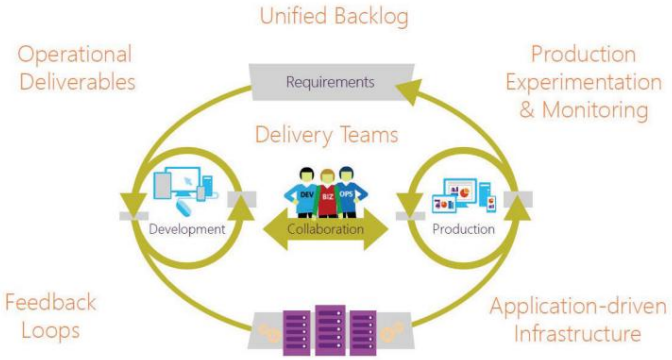
  คุณภาพของซอฟต์แวร์

เป้าหมายของ DevOps

- ปรับปรุงการทำงานร่วมกัน การสื่อสาร

  การบูรณาการและระบบอัตโนมัติระหว่างนักพัฒนา

  และการดำเนินงานด้านไอที



ประโยชน์ที่สำคัญ

•ออกสู่ตลาดได้เร็วขึ้น

- ลดอัตราความล้มเหลวของการออกใหม่
- ปรับปรุงเวลาการฟื้นตัว
- มีเวลาเพิ่มมูลค่าแทนการแก้ไข/

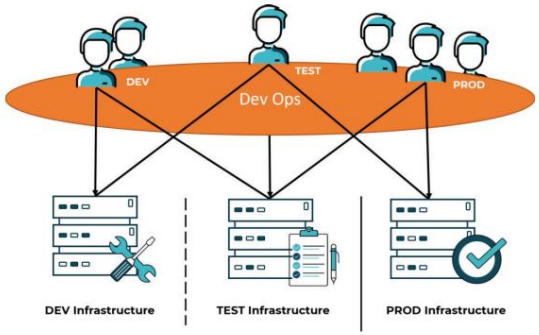
  การบำรุงรักษา.

กลยุทธ์ DevOps

- นักพัฒนาและผู้ดูแลระบบกำลังพูดถึงหมอ

•พวกเขาเป็นทีมและเข้าใจดีว่าทีมไหน

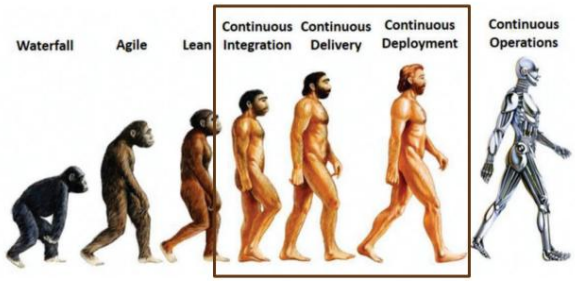
ข้อกำหนดควรได้รับการพิจารณานอกเหนือจากของตนเอง



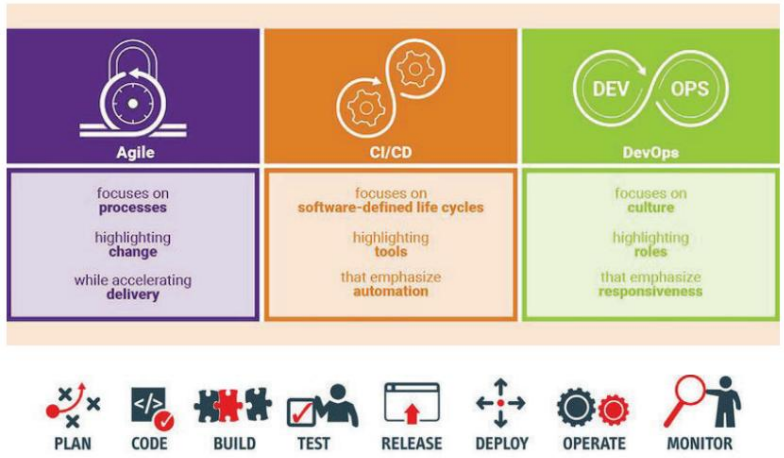
การพัฒนา SDLC

- ลดระยะเวลาในการทดสอบและการปรับใช้

•เติมเต็มช่องว่างใน SDLC สู่ DevOps



Agile, CI/CD และ DevOps

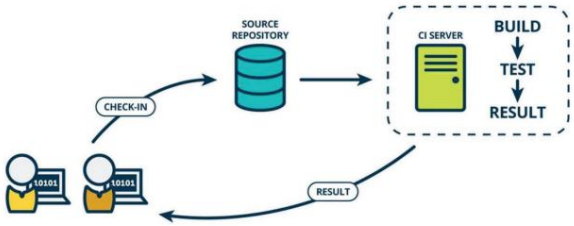


บูรณาการอย่างต่อเนื่อง (CI)

- แนวทางปฏิบัติในการพัฒนาที่กำหนดให้นักพัฒนาต้อง

  รวมโค้ดเข้ากับพื้นที่เก็บข้อมูลที่ใช้ร่วมกันหลายแห่ง

  วันละครั้ง



การส่งมอบอย่างต่อเนื่อง (CD1)

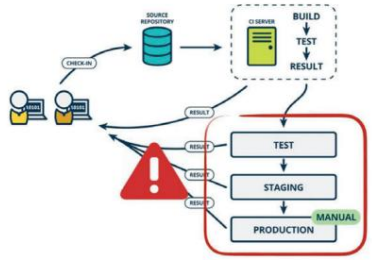
- แนวทางวิศวกรรมซอฟต์แวร์ที่ทีม

  ผลิตซอฟต์แวร์ในระยะเวลาอันสั้น จึงมั่นใจได้ว่า

  สามารถปล่อยวางใจได้ตลอดเวลา

- การปรับใช้งานจริงกับสภาพแวดล้อมการผลิต

  เป็น กระบวนการที่ต้องทำด้วยตนเอง



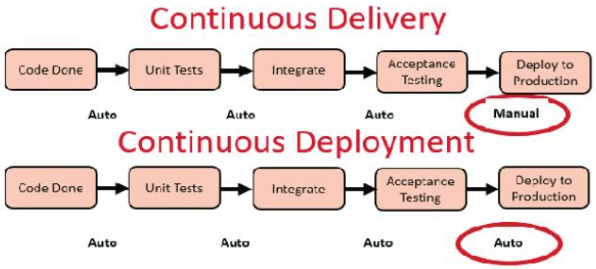
การปรับใช้อย่างต่อเนื่อง (CD2)

- กระบวนการเผยแพร่ซอฟต์แวร์ที่ใช้ ระบบอัตโนมัติ

  การทดสอบ เพื่อตรวจสอบความถูกต้องว่ามีการเปลี่ยนแปลงโค้ดเบสหรือไม่

  ถูกต้องและมั่นคงเพื่อ ความเป็นอิสระ ทันที

  การปรับใช้ กับสภาพแวดล้อมการผลิต



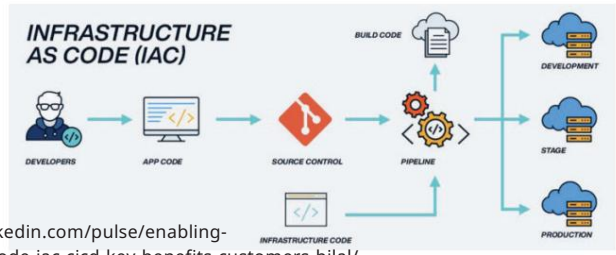
โครงสร้างพื้นฐานเป็นรหัส (IaC)

- IaC คือแนวทางปฏิบัติที่สำคัญของ DevOps ที่เกี่ยวข้องกับ การจัดการ

  และการจัดเตรียมโครงสร้างพื้นฐาน โดยใช้ เครื่องจักร-

  ไฟล์คำจำกัดความที่อ่านได้ แทนการใช้คู่มือ

  การกำหนดค่า



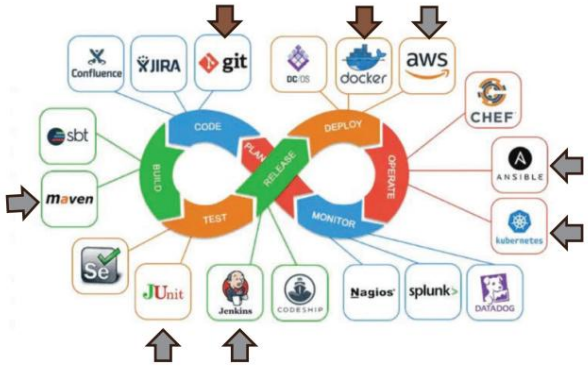
https://www.linkedin.com/pulse/enabling-Infrastructure-code-iac-cicd-key-benefits-customers-bilal/

โครงสร้างพื้นฐานเป็นรหัส (IaC)

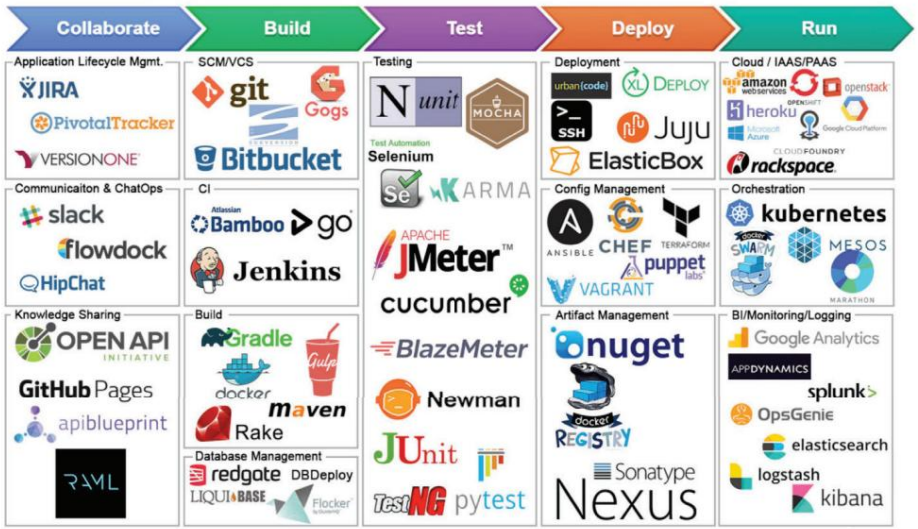
- ประโยชน์ของ IaC: ความเร็ว ความสม่ำเสมอ การย่อขนาด
  ของความผิดพลาดของมนุษย์ ความสามารถในการขยายขนาด
- IaC ช่วยในการพัฒนา การทดสอบ
  และเวิร์กโฟลว์การปรับใช้ในไปป์ไลน์ CI/CD
- IaC สามารถนำไปใช้ได้โดยใช้เครื่องมือเช่น Ansible
  เชฟ หุ่นเชิด และเทอร์ราฟอร์ม

ชุดเครื่องมือเพื่อเสริมความแข็งแกร่งให้กับ DevOps Pipeline

- อันไหนที่คุณรู้จักอยู่แล้ว?
  - ทักษะ bash และ Linux จะเป็นประโยชน์



ชุดเครื่องมือเพื่อเสริมความแข็งแกร่งให้กับ DevOps Pipeline



35 https://www.suntechnologies.com/blogs/ci-cd-pipelines-for-deploying-multiple-projects-best-ways-to-automate-build-and-deployment/

5 อันดับเทรนด์ DevOps ในปี 2023

- เพิ่มการใช้ AI และ ML
  - อัลกอริธึม ML สามารถ วิเคราะห์การเปลี่ยนแปลงโค้ดและ ทำนายได้
    ข้อขัดแย้งและให้คำแนะนำสำหรับ
    ปรับปรุงคุณภาพโค้ด
  - เครื่องมือการปรับใช้ที่ขับเคลื่อนด้วย AI สามารถระบุและ ป้องกันความล้ม
    เหลวในการปรับใช้ ก่อนที่จะเกิดขึ้น
  - อัลกอริธึม ML สามารถ ตรวจสอบแอปพลิเคชัน ได้
    ประสิทธิภาพ ตรวจจับความผิดปกติ และ ปรับปรุง
    ประสบการณ์ผู้ใช้

https://devopscube.com/devops-trends/

5 อันดับเทรนด์ DevOps ในปี 2023

- การขยายตัวของ DevSecOps
  - เมื่อภัยคุกคามทางไซเบอร์พัฒนาขึ้น องค์กรต่างๆ ให้ความสำคัญ กับความปลอดภัยและการปฏิบัติ ตามข้อกำหนดมากขึ้น
  - ให้บริการอย่างครบวงจรและเป็นอัตโนมัติ แนวทางการรักษาความปลอดภัยที่สามารถช่วย ตรวจจับและ แก้ไขช่องโหว่ ในช่วงต้นของการพัฒนา

5 อันดับเทรนด์ DevOps ในปี 2023

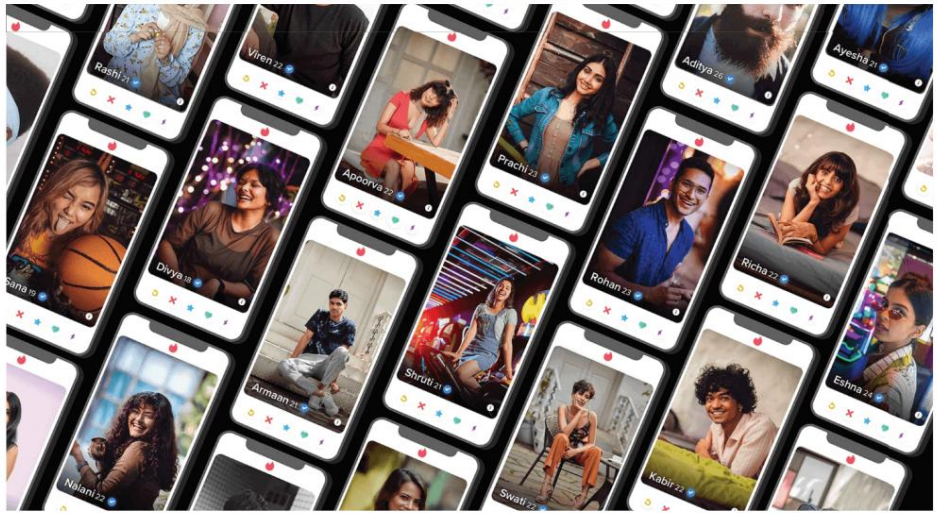
- เทคโนโลยีคลาวด์เนทีฟ
  - Kubernetes, Docker และ การประมวลผล แบบไร้เซิร์ฟเวอร์ เป็นเทคโนโลยีที่ได้รับความนิยมมากที่สุดในการขับเคลื่อนสิ่งนี้ แนวโน้ม
  - เทคโนโลยีเหล่านี้ช่วยให้ทีม DevOps สามารถ สร้างและปรับใช้แอปพลิเคชันได้มากขึ้น ความเร็ว ประสิทธิภาพ และความคล่องตัว

5 อันดับเทรนด์ DevOps ในปี 2023

- DevOps เป็นบริการ (DaaS)
  - DaaS อนุญาตให้บริษัท ต่างๆ การจัดการ โครงสร้างพื้นฐานและกระบวนการ DevOps ช่วยลดภาระการ ปฏิบัติงานของทีมภายใน
  - DaaS มอบแนวทางแก้ไข ปัญหานี้ด้วยการ นำเสนอบริการ DevOps ระดับมืออาชีพแก่องค์กรที่ต้องการว่าจ้างการจัดการ DevOps จากภายนอก
    - การบูรณาการและการส่งมอบอย่างต่อเนื่อง (CI/CD), GitHub การดำเนินการ การจัดการ โครงสร้างพื้นฐานคลาวด์ คอนเทนเนอร์ การตรวจสอบ และความปลอดภัย

5 อันดับเทรนด์ DevOps ในปี 2023

- ให้ความสำคัญกับความสามารถในการสังเกตมากขึ้น
  - ความสามารถในการวัดและ ตรวจสอบประสิทธิภาพ และพฤติกรรมของ แอปพลิเคชันแบบ เรียลไทม์
  - การติดตามแบบกระจาย ช่วยให้ทีม DevOps ติดตามคำขอขณะเดินทางผ่านระบบแบบ กระจายเพื่อ ระบุปัญหาคอขวด และปัญหาอื่นๆ ที่ส่งผลกระทบต่อประสิทธิภาพการ ทำงาน
  - การวิเคราะห์บันทึก จะระบุรูปแบบและความผิดปกติใน เพื่อ ระบุสาเหตุของปัญหา และทำการปรับปรุงแอปพลิเคชันตามเป้าหมาย



tinder

การออกแบบระบบ- Tinder | ต้นทุนในการพัฒนา | วิธีหารายได้

https://dev.to/mukulalpha/system-design-tinder-cost-to-develop-how-to-earn-revenue-4mno

แอปพลิเคชั่นหาคู่

tinder

- คนหนุ่มสาวยุคใหม่ยอมรับการ

  ไลฟ์สไตล์การออกเดท ส่วนการหาคู่ออนไลน์

  ยังคงเติบโต

•Tinder เป็นตัวอย่างที่ดีว่าแอปหาคู่สามารถทำได้อย่างไร

  จะกลายเป็นธุรกิจที่ทำกำไรได้สูง

  - อัตราการแปลงการจับคู่สูง
  - คุณสมบัติที่ใช้งานง่าย

ข้อกำหนดด้านการทำงานขั้นพื้นฐาน

tinder

•เข้าสู่ระบบได้หลายวิธี เช่น Facebook, Instagram,

  google หมายเลขโทรศัพท์ และอื่นๆ: อนุญาตให้คุณสมัครใช้งาน

  โดยใช้ ID Facebook, Instagram ID และอื่นๆ ของคุณ

- ผู้ใช้ทุกคนจะต้อง มีโปรไฟล์ผู้ใช้ของตนเอง ซึ่งรวมถึงอายุ ระยะทาง เพื่อนร่วมกัน

  ความสนใจร่วมกัน และคำอธิบายเป้าหมาย

•ตำแหน่งทางภูมิศาสตร์ตรวจจับ ตำแหน่งปัจจุบัน ของผู้ใช้

- ผู้ใช้ควรจะสามารถ ดูคำแนะนำ ในก

  ภูมิภาคทางภูมิศาสตร์

ข้อกำหนดด้านการทำงานขั้นพื้นฐาน

tinder

- ค้นหา โดยการกรอง (ได้แก่ อายุ เพศ ระยะทาง และอื่นๆ

  บน)

- การปัด (การนำเสนอคุณค่าที่ไม่ซ้ำใคร) คือการกระทำที่แสดงหมวดหมู่เป้าหมายผู้ใช้

  ขวาหมายถึงเหมือนใครบางคน และซ้ายหมายถึงปฏิเสธใครบางคน

- จับคู่: ผู้ใช้ทั้งสองสามารถเริ่มแชทได้ครั้ง เดียว

  ผู้ใช้ปัดไปทางขวากัน

- แชท: ข้อความส่วนตัวกับผู้ใช้รายอื่น
- การแจ้งเตือนแบบพุช: แจ้งเตือนผู้ใช้เมื่อได้รับข้อความใหม่

  ส่งข้อความถูกใจหรือจับคู่หรือแชท

มาตราส่วน

- สมาชิกมากกว่า 57 ล้านคน
- ผู้ใช้งานมากกว่า 12 ล้านคน
- 55+ พันล้านแมตช์
- ดาวน์โหลดมากกว่า 100 ล้านครั้ง
- 1.8 พันล้านครั้งปัดทุกวัน (ปัดซ้าย, ขวา ปัดเลย ชอบมาก)
- รองรับภาษามากกว่า 40 ภาษา

สถาปัตยกรรมการปรับใช้สำหรับ

สถาปัตยกรรมการปรับใช้

Tinder ได้รับการโฮสต์บน AWS Cloud โดยสมบูรณ์ ไม่มีเว็บแอปพลิเคชันใด ๆ ยกเว้น IOS และ Andriod Tinder ใช้ AWS amplify เพื่อสร้างและทดสอบแอปพลิเคชันมือถือ, MongoDB สำหรับ DB และ Redis สำหรับแคชและฐานข้อมูลในหน่วยความจำ

สถาปัตยกรรมเชื้อจุดไฟ

- การออกแบบระดับสูง

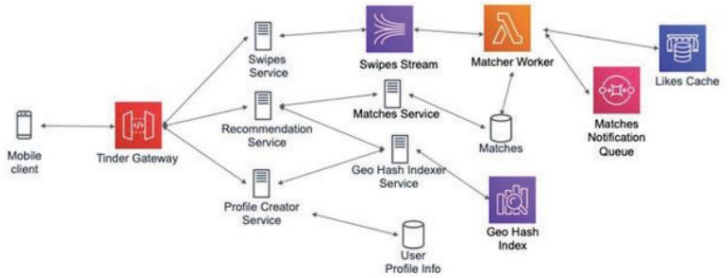


Fig: Tinder Architecture

สถาปัตยกรรมเชื้อจุดไฟ

- ไมโครเซอร์วิสจำนวนมากจะอยู่ด้านหลังเกตเวย์เพื่อให้บริการตามคำขอของผู้ใช้
- บริการสร้างโปรไฟล์จะถูกเรียกใช้เมื่อมีการสร้างโปรไฟล์ผู้ใช้
- ผู้ใช้จะถูกเพิ่มลงในดัชนีการแบ่งส่วนทางภูมิศาสตร์ที่เกี่ยวข้องเพื่อให้
  ผู้ใช้จะปรากฏตามคำแนะนำของผู้ใช้ใกล้เคียง
- บริการแนะนำจะสอบถามดัชนีนี้เมื่อได้รับคำขอ
  เพื่อสร้างคำแนะนำสำหรับผู้ใช้รายอื่น
- เมื่อผู้ใช้เริ่มปัดตามคำแนะนำ ก็ให้ปัดนิ้ว
  บริการจะได้รับการปัดและวางลงในสตรีมข้อมูล (เช่น AWS Kinesis/SQS)

- กลุ่มคนงานจะอ่านข้อมูลจากสตรีมเหล่านั้นเพื่อสร้างการจับคู่
- ผู้ปฏิบัติงานทำได้โดยการสืบค้น LikesCache เพื่อตรวจสอบว่าตรงกันหรือไม่
- หากเป็นการจับคู่ การแจ้งเตือนการจับคู่จะถูกส่งไปยังผู้ใช้ทั้งสองที่ใช้
  เทคโนโลยีเช่น WebSockets

เครื่องยนต์แนะนำ

คุณสมบัติ

- เวลาแฝงต่ำ: เมื่อมีคนลงชื่อเข้าใช้แอปพลิเคชัน เราจำเป็นต้องโหลดโปรไฟล์/โปรไฟล์ที่ตรงกันที่มี
  ศักยภาพอย่างรวดเร็ว
- ไม่ใช่เรียลไทม์: ไม่เป็นไรถ้าไม่ใช่เรียลไทม์ เช่น ถ้ามีคนใหม่
  เข้าร่วม tinder ไม่เป็นไรหากต้องใช้เวลาสักครู่เพื่อแสดงโปรไฟล์ของบุคคลนี้
  ในบัญชีของเรา
- แบ่งส่วน/แจกจ่าย ได้ง่าย : เนื่องจากเรามีโปรไฟล์มากมายจากทั่วโลก เครื่องมือแนะนำนี้จึงควร
  สามารถแบ่งย่อยข้อมูลได้ เนื่องจากเราไม่สามารถเก็บไว้ในระบบเดียวได้

- ค้นหาข้อความแบบเต็ม: เราจำเป็นต้องค้นหาให้ทั่วทั้งโปรไฟล์ของ
  บุคคลเพื่อให้คำแนะนำที่ดีขึ้น
- อินเทอร์เฟซ HTTP: หรือเว็บซ็อกเก็ตเพื่อรับข้อมูลและส่งไปที่
  แอปพลิเคชัน.
- โครงสร้างข้อมูล: XML/JSON

Elasticsearch สามารถบรรลุการตอบสนองการค้นหาที่รวดเร็ว
เนื่องจากแทนที่จะค้นหาข้อความโดยตรง กลับค้นหาดัชนีแทน

กลยุทธ์การแนะนำ

•Tinder โดยพื้นฐานแล้วต้องการให้ผู้คนได้พบปะกัน

•หากฉันเป็นผู้ใช้จากสถานที่ X อินเดีย ฉันจะทำ
เห็นได้ชัดว่าชอบที่จะจับคู่กับใครสักคน
ซึ่งมาจากตำแหน่ง X + 50 กม.

- จะทำให้ latency ต่ำในการค้นหาแบบยืดหยุ่นได้อย่างไร?
  - แบ่งข้อมูลตามที่ตั้งทางภูมิศาสตร์!!

•...ยังไง?

- คำขอที่มาจากกล่องจะให้บริการโดยเซิร์ฟเวอร์ในกล่องนั้น

- กล่องที่มีความหนาแน่นของประชากรสูงอาจต้องการมากกว่านั้น
  เซิร์ฟเวอร์หนึ่งเครื่อง

- ขนาดของกล่องในพื้นที่ต่างๆจะถูกกำหนดโดย
  จำนวนผู้ใช้ที่ไม่ซ้ำ จำนวนผู้ใช้ที่ใช้งานอยู่ และจำนวนคำค้นหาจากภูมิภาคเหล่านี้

- เมื่อมีคนเปิด Tinder โทรศัพท์ของพวกเขาจะถามคำถาม
  สู่ระบบผู้ทำแผนที่

- ระบบผู้ทำแผนที่ใช้ละติจูดและลองจิจูดของผู้ใช้
  เพื่อกำหนดว่าข้อมูลของพวกเขาถูกเก็บไว้บนเซิร์ฟเวอร์ใด

- เซิร์ฟเวอร์นี้เป็นเซิร์ฟเวอร์เดียวกันกับที่เก็บข้อมูลของผู้ใช้
  การแข่งขันที่อาจเกิดขึ้น

- ระบบ mapper สามารถอยู่ในตำแหน่งทางกายภาพใดก็ได้
  ที่ตั้ง.

- ข้อมูลทั้งหมดสำหรับเซลล์ใดเซลล์หนึ่งจะอยู่ในเซลล์นั้น
  เซิร์ฟเวอร์

- ข้อมูลสำหรับเซลล์ 1, 2, 3, 4, 5, 6 และ 7 จะถูกจัดเก็บไว้ในเซิร์ฟเวอร์
  1, 2, 3, 4, 5, 6 และ 7 ตามลำดับ

- คุณเป็นผู้ใช้ Tinder ที่อยู่ในเซลล์ 3 และได้ตั้งค่าช่วงของคุณเป็น
  100 กม.

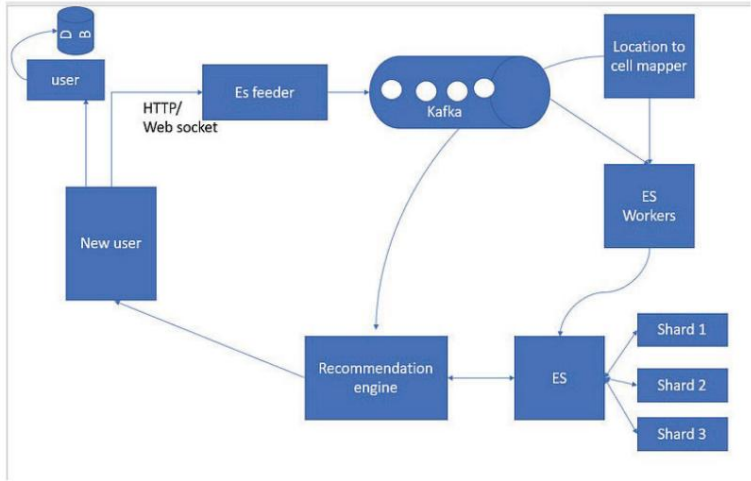
- ซึ่งหมายความว่าคุณต้องการทราบการแข่งขันที่เป็นไปได้ทั้งหมดของคุณ
  ภายใน 100 กม. จากที่ตั้งของคุณ

- ข้อมูลของคุณอยู่บนเซิร์ฟเวอร์ 3

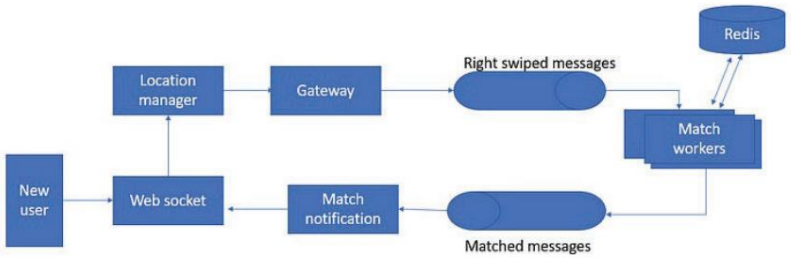
- ข้อมูลของการแข่งขันที่เป็นไปได้จะอยู่ในรัศมี 100 กม. ซึ่งรวมถึงเซลล์ 1 ถึง 7

- เมื่อคุณเปิด Tinder คำขอของคุณจะถูกส่งไปยังเซิร์ฟเวอร์ทั้งหมด เช่น
  เซิร์ฟเวอร์ 1 ถึง 7

- เซิร์ฟเวอร์จะรวบรวมคำแนะนำ
  และส่งพวกเขากลับมาหาคุณ



- มีการแข่งขันนับล้านรายการเกิดขึ้นทุกวัน
- เราสามารถมีบริการจัดหาคู่ได้หนึ่งบริการต่อเซลล์หรือไม่กี่กลุ่ม
  เซลล์พร้อมกับบริการจับคู่เดียว
- จะมีบริการจับคู่บางอย่างที่ทำงานในเวลาเดียวกัน
  ถึงเวลาที่จะปรับสมดุลระหว่างการค้นหาตามสถานที่ตั้ง
- บริการค้นหาคู่แต่ละรายการจะอยู่ในเซลล์ไม่กี่เซลล์แทน
  เพียงหนึ่งเซลล์
- การแข่งขันจะไม่เกิดขึ้นระหว่างประเทศ แต่จะเกิดขึ้นใน
  เซลล์ที่แนะนำโปรไฟล์ให้กับผู้ใช้
- ตัวอย่างเช่น หากเราแนะนำโปรไฟล์ 100 โปรไฟล์ให้กับผู้ใช้ มีโอกาสดีที่โปรไฟล์เพียง 20-30 โปรไฟล์เท่านั้น
  ที่จะถูกปัดไปทางขวา



- เมื่อผู้ใช้ปัดไปทางขวา ข้อความจะถูกส่งไปยังการจับคู่
  บริการ.
- ผู้จัดการสถานที่จะเป็นผู้กำหนดว่าชิ้นส่วนหรือบริการจับคู่ใด
  ข้อความนี้ควรไปที่
- จากนั้นข้อความจะเปลี่ยนเส้นทางไปยังเกตเวย์ซึ่งเชื่อมต่ออยู่
  คาฟคา.
- ขณะนี้ข้อความอยู่ในคิวแล้ว
- จะมีบริการจับคู่ตั้งแต่หนึ่งบริการขึ้นไปที่ข้อมูลนี้จะถูกถ่ายทอดไป ทั้งนี้ขึ้นอยู่กับจำนวนชิ้นส่วน
- ข้อมูลที่บันทึกไว้ในที่นี้คือใครปัดไปทางขวากับใคร ตำแหน่งของพวกเขา และข้อมูลเมตาอื่นๆ
- อาจมีพนักงานแบบขนานที่คอยอ่านข้อความจาก
  คิวคาฟคา.

- หาก A ปัดไปทางขวาบน B รายการเช่น "A_B" จะถูกเพิ่มเข้าไป
  เรดิส
- เมื่อ B ปัดไปทางขวาบน A กระบวนการเดียวกัน
  เกิดขึ้น
- เจ้าหน้าที่จับคู่รับข้อความและตรวจสอบ Redis เพื่อดูว่า A เคยปัดไปทางขวาบน B หรือไม่
- หากมี หมายความว่ามีการแข่งขันเกิดขึ้น
  และข้อความจะถูกเพิ่มลงในคิวที่ตรงกัน
- การแจ้งเตือนการจับคู่จะรับข้อความนี้และส่งไปยังทั้ง A และ B ผ่านทาง WebSockets

อ่านเพิ่มเติมเกี่ยวกับ

- https://interviewnoodle.com/tinder-system-สถาปัตยกรรม-2012902cb415
- https://medium.com/system-design-concepts/dating-การออกแบบระบบแอปพลิเคชัน-aae411412267
- https://www.techaheadcorp.com/blog/understand-สถาปัตยกรรมการออกแบบระบบของ Tinder/
- http://highscalability.com/blog/2022/1/17/designing-เชื้อจุดไฟ.html

อ้างอิง

- https://www.jitendrazaa.com/blog/salesforce/continuous-integration-vs-continuous-delivery-vs-continuous-deployment/
- https://devopscube.com/devops-trends/
- https://dev.to/pavanbelagatti/learn-how-to-setup-a-cicd-pipeline-จาก-scratch-for-a-go-application-4m69
- https://www.hostinger.com/tutorials/ssh/basic-ssh-คำสั่ง
- https://www.hostinger.com/tutorials/ssh-tutorial-how-ไม่-ssh-ทำงาน