**ระบบเก่าที่เป็น Monolith !!**

เป็นระบบที่พัฒนาด้วย Node.js ตั้งแต่ปี 2012  
แน่นอนว่า  
ยิ่งนานวันระบบยิ่งมีขนาดใหญ่ขึ้น  
ยิ่งนานวันสมาชิกของทีมพัฒนายิ่งมากขึ้น  
ยิ่งนานวันความซับซ้อนของระบบยิ่งยุ่งเหยิงขึ้น  
การดูแลรักษารวมทั้ง operation ต่าง ๆ ยิ่งยากขึ้นมา  
และหลาย ๆ ส่วนการทำงานเริ่มเกิดปัญหาคอขวด  
ดังนั้นในปี 2018 ทางทีมพัฒนาจึงเริ่มนำแนวคิดของ Microservices มาใช้งาน  
เพื่อลดปัญหาต่าง ๆ ที่พบเจอ  
มาดูกันว่าขั้นตอนการนำแนวคิด Microservices เป็นอย่างไรบ้าง  
ซึ่งนับว่าเป็นประสบการณืที่น่าศึกษาเป็นอย่างมาก

**ก่อนอื่นทีมต้องเข้าใจก่อนว่า Microservices คืออะไร ?**

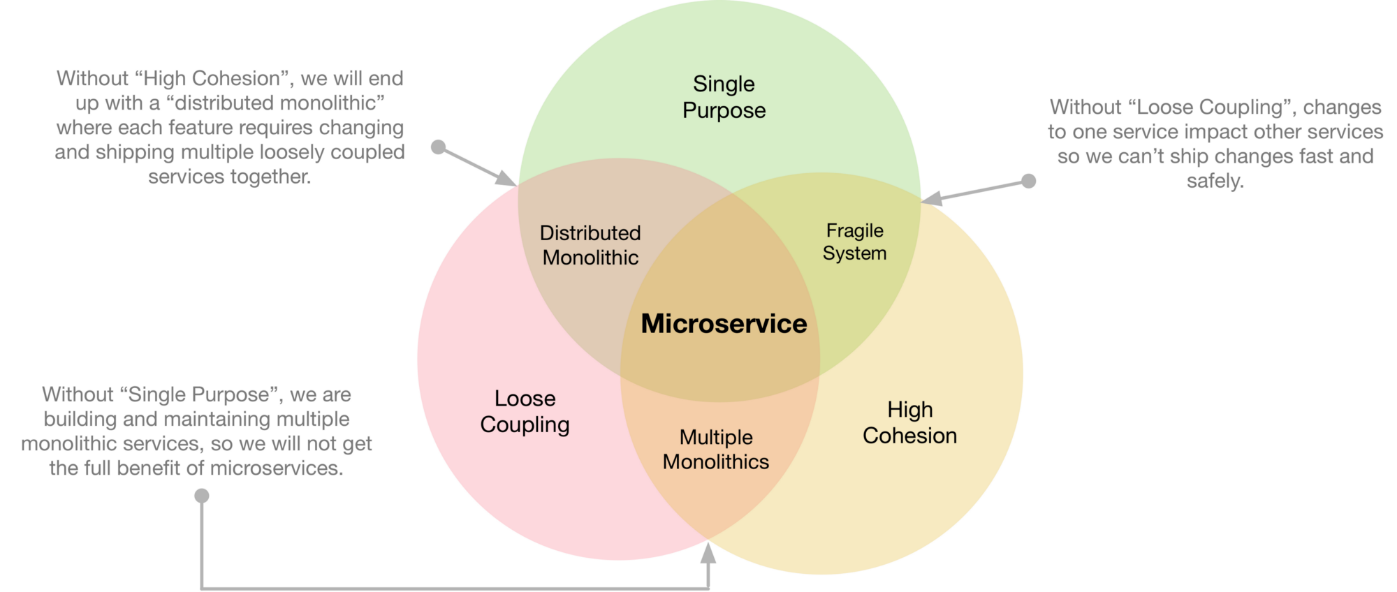
เพื่อทำให้ทุกคนมีความเข้าใจตรงกัน  
เพราะว่าเป็นคำที่พูดถึงเยอะมาก และสร้างความสับสนอย่างมาก  
ประกอบไปด้วย 2 เรื่องคือ

**เรื่องที่ 1 อะไรคือ Microservices**

ในบทความจะยึดแนวปฏิบัติ 3 ข้อคือ

1. **Single purpose** แต่ละ service มีเป้าหมายการทำงานเดียว และ ทำให้มันดี หรือจบในตัวเอง
2. **Loose coupling** แต่ละ service ควรตัดขาดหรือรู้จักแต่ละ service เพียงนิดเดียว มันจะช่วยลดผลกระทบเมื่อ service ต่าง ๆ มีการเปลี่ยนแปลง รวมไปถึงการติดต่อสื่อสารระหว่าง service ควรเกิดขึ้นผ่านทาง public service interface เท่านั้น
3. **High cohesion** แต่ละ service ต้องซ่อนทั้ง behaviour และ data ที่เกี่ยวข้องหรือใช้งานไว้ในตัวเอง ดังนั้นถ้าต้องการสร้าง service ขึ้นมาใหม่ก็ควรที่จะเปลี่ยนภายใน service เท่านั้น ไม่ควรไปกระทบ service อื่น ๆ มิเช่นนั้น service ต่าง ๆ จะกลายเป็น distributed monolith system ไปนะ ต้องระวังด้วย

แสดงความสัมพันธ์ดังรูป

[](https://medium.engineering/microservice-architecture-at-medium-9c33805eb74f)

**เรื่องที่ 2 อะไรบ้างที่ไม่ใช่ Microservices**

ไม่ใช่ service ที่มีจำนวน Line of Code น้อย ๆ หรือ ไม่ใช่ micro task  
ไม่ใช่ service ที่พัฒนามาจาก teachnology ใหม่ ๆ เสมอ แต่ microservices อนุญาตให้ทีมได้ลองของใหม่ ๆ ได้ง่ายเท่านั้นเอง และจะดีมากถ้าพัฒนาด้วย technology เดิม เพราะว่าทีมจะได้ประโยชน์จากการแยก service เป็นอย่างมาก  
ไม่ใช่ service ที่สร้างมาใหม่ตั้งแต่เริ่มต้น  
สิ่งที่ควรเกิดขึ้นคือ  
ถ้าระบบเดิมของเราที่เป็น monolith นั้น  
ทำการแยกเป็น modular ที่ดีอยู่แล้ว  
มันจะง่ายต่อการแยกออกมาเป็น service เล็ก ๆ โดยไม่ต้องพัฒนาใหม่เลย  
ดังนั้นอะไรก็ตามทีไม่ใช่ Microservices แสดงว่า จะไม่เป็นไปตามแนวคิด 3 ข้อจากข้างต้นนั่นเอง

**เรื่องที่น่าสนใจอีกคือ ทำไมต้องทำตอนนี้ด้วยละ ?**

เพราะว่า เมื่อใดก็ตามที่มีการถามคำถามนี้แล้ว  
มันจะเกิดข้อจำกัดมากมายขึ้นมา  
ทั้งผลกระทบที่จะเกิดขึ้น ?  
ทั้งค่าใช้จ่ายต่าง ๆ ?  
ทั้ง overhead จากงานแทรกต่าง ๆ ?  
และเรื่องอื่น ๆ อีกมากมาย !!  
แต่ข้อดีของคำถามนี้คือ มีคำตอบเยอะมาก  
ดังนั้นก็ทำการจัดเรียงความสำคัญมันซะ !! ยากนะแต่ต้องทำ

**เหตุผลหลัก ๆ ของทาง Medium.com ที่ต้องลงมือทำนั้น**  
เนื่องจากระบบเดิมที่พัฒนาด้วย Node.js นั้นมีขนาดใหญ่  
กลายเป็นปัญหาคอขวด  
ทั้งเรื่องของ performance ที่ Node.js จะมีปัญหาเมื่อมีการใช้ I/O สูงมาก ๆ  
ดังนั้นการ scale ระบบใหญ่ ๆ ถูกพิสูจน์แล้วว่า  
วิธีการต่าง ๆ ไม่ทำให้ดีขึ้นเลย  
ยิ่งระบบใหญ่ขึ้น feature มากขึ้น ทีมก็ส่งมอบงานได้ช้าลง ประสิทธิภาพก็แย่ลงไปเรื่อย ๆ

**ยังไม่พอนะ**  
เมื่อพัฒนา feature ใหม่ ๆ มันต้องกระทบกับส่วนอื่น ๆ ไปหมด  
ไม่สามารถย้ายหรือลบส่วนงานบางส่วนออกไปได้  
เพราะว่าไปกระทบส่วนอื่น ๆ  
ทีมไม่กล้าที่จะแก้ไขหรือทำการเปลี่ยนแปลงระบบที่มีขนาดใหญ่ได้  
เพราะว่าไม่สามารถระบุผลกระทบที่เกิดขึ้นได้เลย  
ส่งผลต่อการพัฒนาระบบงานอย่างมาก  
ยิ่งทำยิ่งช้า

**ส่วนการ scale ระบบยิ่งไม่ต้องพูดถึงยากมาก ๆ**  
รวมทั้ง task ต่าง ๆ ที่ใช้ในการจัดการ resource  
บาง feature มี task ที่ง่าย  
บาง feature มี task ที่ซับซ้อน  
บาง feature ใช้ resource มากน้อยแตกต่างกัน

**ยังมีเรื่องอื่น ๆ อีกมากมาย**  
ยกตัวอย่างเช่น ทีมไม่สามารถทดลอง technology ใหม่ ๆ ได้  
เพราะว่าต้องผูกมัดกับ technology เก่าให้ได้ ซึ่งยากมาก ๆ