Chapter 2 Introduction to Microservices

Monolith

Monolith คือ architectural style แบบหนึ่ง ที่มอง Application เป็น single executable component

- ข้อดี >> **Maintainability**, **Testability**, **Deployment**
- ข้อเสีย >> **Monolithic hell** >> Application โตเกินกว่าที่จะ handle ได้
  (Agile ไม่ประสบความสำเร็จ และข้อดีทั้งสามข้อจะไม่เป็นจริงแล้ว)
  1. เพิ่มความซับซ้อนต่อ developers ในการพัฒนา >> Bugfix ยาก
  2. Development ทำได้ช้า >> Application มีขนาดใหญ่ IDE compile ทีใช้เวลาเยอะมาก
  3. กว่าจะ commit ไปจนถึง deploy ใช้เวลานาน >> Merge ยาก
  4. Scalable ได้ยาก >> ทำได้แค่ vertical scaling
  5. ส่งมอบ reliable monolith เป็นเรื่องยาก >> เพราะการทดสอบให้ครบถวนก็เป็นเรื่องยาก (ภายในเวลาที่เร่ง ๆ อยู่เป็นปกติ)
  6. Obsolete technology (ล้าสมัย) >> การ upgrade version หรือเปลี่ยน framework/language เป็นเรื่องที่ถ้าทำแล้ว อาจจะต้องทำใหม่ทั้งหมด

Microservices

Microservices คือ

- architectural styles ที่แบ่ง application ออกเป็น services ย่อย ๆ โดยคำนึงถึง Business capability เป็นหลัก
- component ที่สามารถ deploy ได้อย่างเป็นอิสระต่อกัน

Scale Cube and Microservices

- **X-axis (Horizontal Duplication Scale by Cloning)**
  - One instance >> Many instance
  - มี identical instances หลาย ๆ ตัวที่มีข้อมูลเก็บไว้เหมือนกัน แล้วช่วยกันทำงานโดยกระจายจาก **load balancer**
  - เป็นวิธี common ของการ scaling ใน monolithic application
- **Y-axis (Functional Decomposition Scale)**
  - เป็นแกนเดียวที่ช่วยให้ Monolith >> Microservices
  - แตก Application ออกเป็น services ย่อย ๆ
- **Z-axis (Data Partitioning)**
  - One Partition >> Many partition
  - มีการทำ partition โดยใช้ field บาง field เช่น user ID ในการระบุว่า request นี้ให้ไปทำที่ไหน โดยกระจายจาก **Router**

DevOps and Organization

- **DevOps** >> Microservices เพิ่มความสามารถในด้าน testability, deployability
- **Organization** >> Microservices ทำให้ทีมมีความคล่องตัว และแต่ละทีมมีความเป็นอิสระต่อกัน (autonomous)
- **New Technology** >> หากอนาคต อยากสร้าง service ใหม่ที่ใช้ technology ที่ต่างจากเดิม สามารถทำได้

Advantages & Disadvantages of Microservices

Advantages

1. **Enables the CD** in large and complex application
   - เพราะ microservices เพิ่มประสิทธิภาพในด้าน testability, deployability, autonomous and loosely coupled
2. **Small** and **easily maintained**

3. **Independently deployable**
4. Enable team to be **autonomous**
5. Better **fault isolation**
6. Easily adopt **new technology**

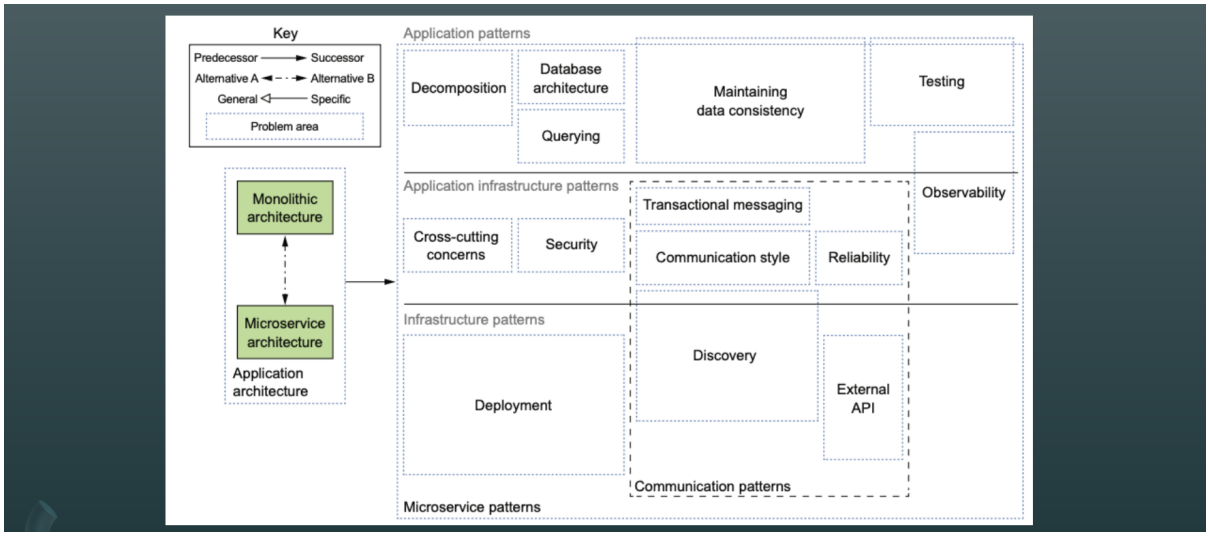
Disadvantages

- จะตัดสินใจว่ามี microservices หรือไม่นั้นไม่ใช่เรื่องง่าย
  - Start-up ช่วงเริ่มต้นเหมาะกับ monolith มากกว่า คำถามคือเมื่อไหร่จะทำเป็น microservices
- การจะแบ่งออกว่ามีกี่ microservices ไม่ใช่เรื่องง่าย
  - แบ่งผิด >> ได้ distributed monolith
- Distributed system มีความซับซ้อน จัดการได้ยาก ต้องมีความรู้เรื่อง technology
- ต้องมี Coordination ที่ดีระหว่าง service
  - จากการที่หากอันนึงพัง อันอื่นอาจจะได้รับผลกระทบไ

Microservices Pattern Language

เป็นภาษาที่ใช้บรรยาย microservices >> ว่า architecture นั้น ๆ ต้องบรรยายอะไรบ้าง

- ช่วยให้เราตัดสินใจได้ดี เพราะบางอย่างก็มีคนทำมาก่อน และแก้ไขปัญหาไปแล้ว (ตอนเลือก option เลือกดี ๆ เพราะแต่ละ pattern มี **trade-off)**



- **Predecessor** คือสิ่งที่ต้องทำหรือตัดสินใจก่อน **Successor**
- **Alternative A** กับ **Alternative B** ต้องเลือกทำอันใดอันหนึ่ง
- **General** เป็นภาพกว้างกว่าของ **Specific**
- **Problem Area** เป็นพื้นที่ปัญหาหรือประเด็นอื่น ๆ ที่ต้องดูหรือตัดสินใจต่อ ๆ ไป

Summary

- SW Architecture ทำขึ้นเพื่อแสดงและตอบโจทย์ NFR
- เพื่อให้เกิด CD เราควรใช้ architectural style ให้เหมาะสม
  - Small app >> Monolith
  - Complex app >> Microservice
- **Pattern Language** ช่วยในการตัดสินใจ ควรใช้ แต่ก็ดู Trade-off ดี ๆ