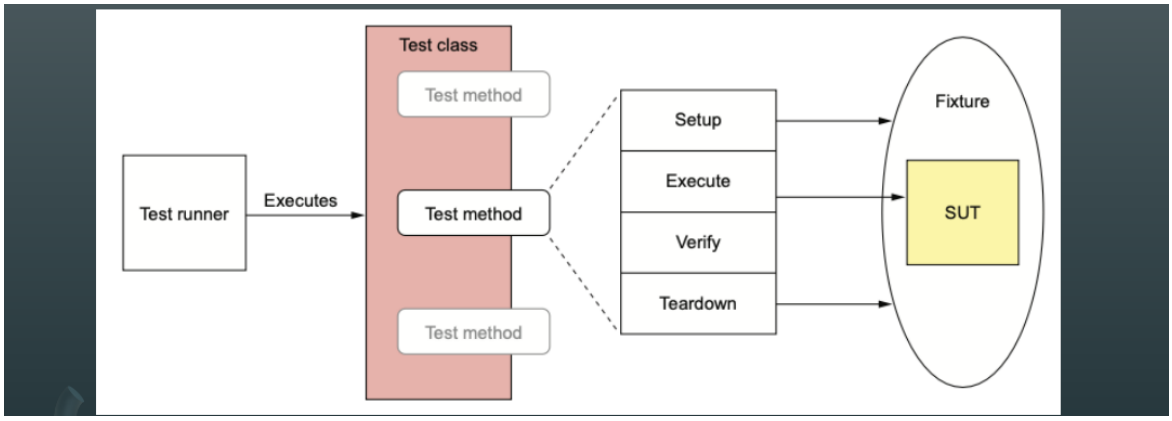
Chapter 09 Testing Microservices

Overview of Testing Microservices

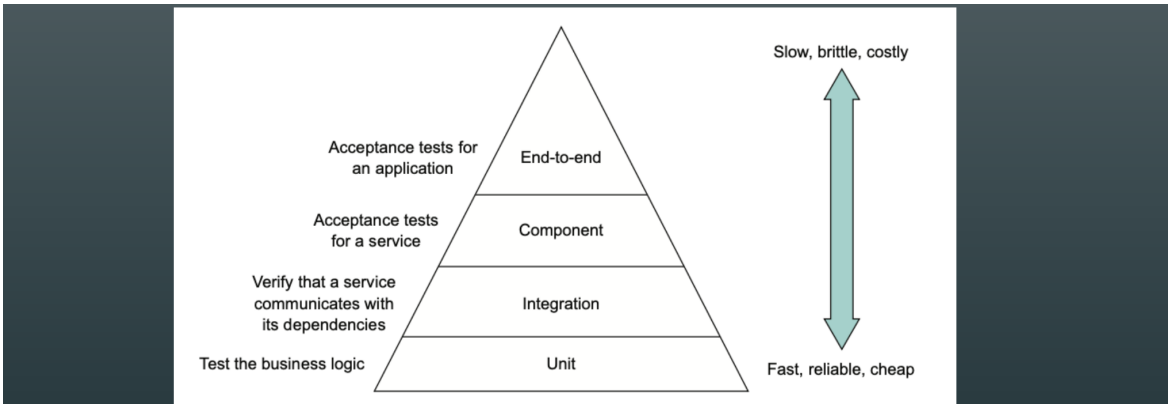
- **Manual testing** >> inefficient & too late in delivery process
- **Ideally workflow** >> edit code → run test → repeat

Testing Microservices

- **System Under Test (SUT)** >> เป็น class หรือ application ที่เป็นเป้าหมายในการ test
- **Test Suit** >> collection of related test
- Writing **automated test** require 4 phases
  1. Setup >> เริ่ม setup SUT, dependencies หรือ initial state
  2. Exercise >> invoke SUT
  3. Verify >> ดูผลลัพธ์ของ invocation SUT และ final state
  4. Teardown >> (optional) ปรับค่าบางอย่างให้กลับไปเป็นเหมือนเดิมกับตอน initial state



- Test Using **Mock** and **Stub** >> replace dependencies with **test double** (ซึ่งมี 2 ประเภท)
  1. **Mock**
  2. **Stub** >> return value to SUT
- **Types of Test** (focused on automated test on FR)
  1. Unit test >> small part (class)
  2. Integration test >> between service and infrastructure (also among services)
  3. Component test >> acceptance test for individual service
  4. End-to-End test >> acceptance test for entire application



Consumer-Driven Contract Testing

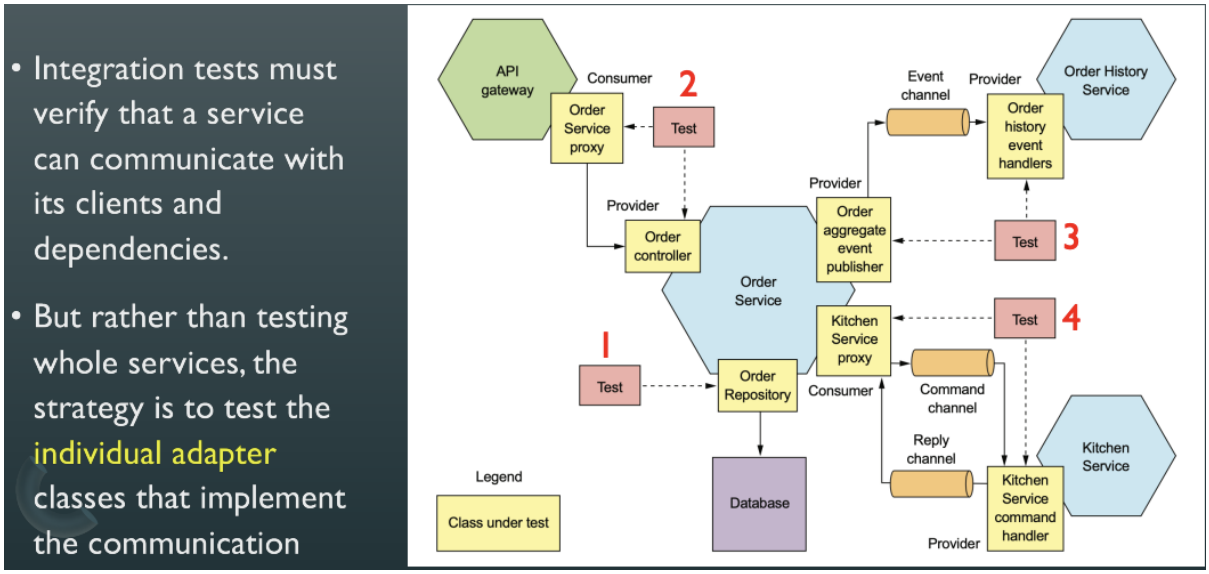
จากการที่ใน Microservices มี IPC หลากหลายแบบ

- **Consumer Contract Test** >> Verify shape of provider'API meets consumer expectation or not
- **Contract Test Suit** >> สิ่งที่ consumer อยากได้ ซึ่งจะใช้ในการทดสอบระบบว่าตอบโจทย์ความต้องการหรือไม่
  - Provider และ Consumer ต้องออกแบบและใช้งานตามที่ระบุ

Writing Unit Test

- **Unit test** >> ใช้เพื่อยืนยันว่า class มีพฤติกรรมดังที่คาดหวังหรือไม่
- **Solitary Unit Test** >> ทดสอบ controller และ service
- **Sociable Unit Test** >> ทดสอบ domain object เช่น entity, value object, saga

Writing Integration Test



Integration Test >> service communicates กับ dependencies และระหว่าง services ได้ดี ซึ่งแทนที่จะทดสอบทั้งหมด ก็ทดสอบ **Individual Adaptor** ว่าสามารถทำงานได้ถูกต้องหรือไม่

1. **Persistence Integration tests** >> ทดสอบว่า database access logic ถูกไหม
   a. setup → execute → verify → teardown
2. **Integration testing REST style interactions** >> ทดสอบ REST
   a. ใช้ Contract เพื่อตรวจสอบ Controller และ Proxy

3. **Integration testing publish/subscribe-style interactions** >> ทดสอบ event subscribing
    a. ใช้ Contract เพื่อตรวจสอบ Publisher และ Handler
4. **Integration tests for asynchronous request/response interactions** >> ทดสอบ async
    a. ใช้ Contract เพื่อตรวจสอบ Command Channel และ Reply Channel

Developing Component Test

- **Component Test** >> การทำ **Acceptance Test** ที่มอง **Service** เป็น **Blackbox** โดยสนใจพฤติกรรมเป็นหลัก
    - เปลี่ยน dependencies → stub

Defining Acceptance Test

- เป็น **Business-facing test** ที่ derived มาจาก user stories และ use case
- **Gherkin** >> Specification (English-like scenario)
- **Cucumber** >> Automation Framework

Writing End-to-End Test

- เป็น **Business-facing test** ต่างจาก Component Test คือทดสอบ Multiple Actions (ทั้งระบบ)
- ไม่ใช่ Stub แล้ว ใช้ทุกอย่างของจริงหมด
- ช้า และใช้ทรัพยากรเยอะ