Testing Microservices

Memi Lavi
www.memilavi.com

Testing Microservices

- Testing is important in all systems and all architecture types
- With Microservices it’s even more important
- Testing Microservices poses additional challenges

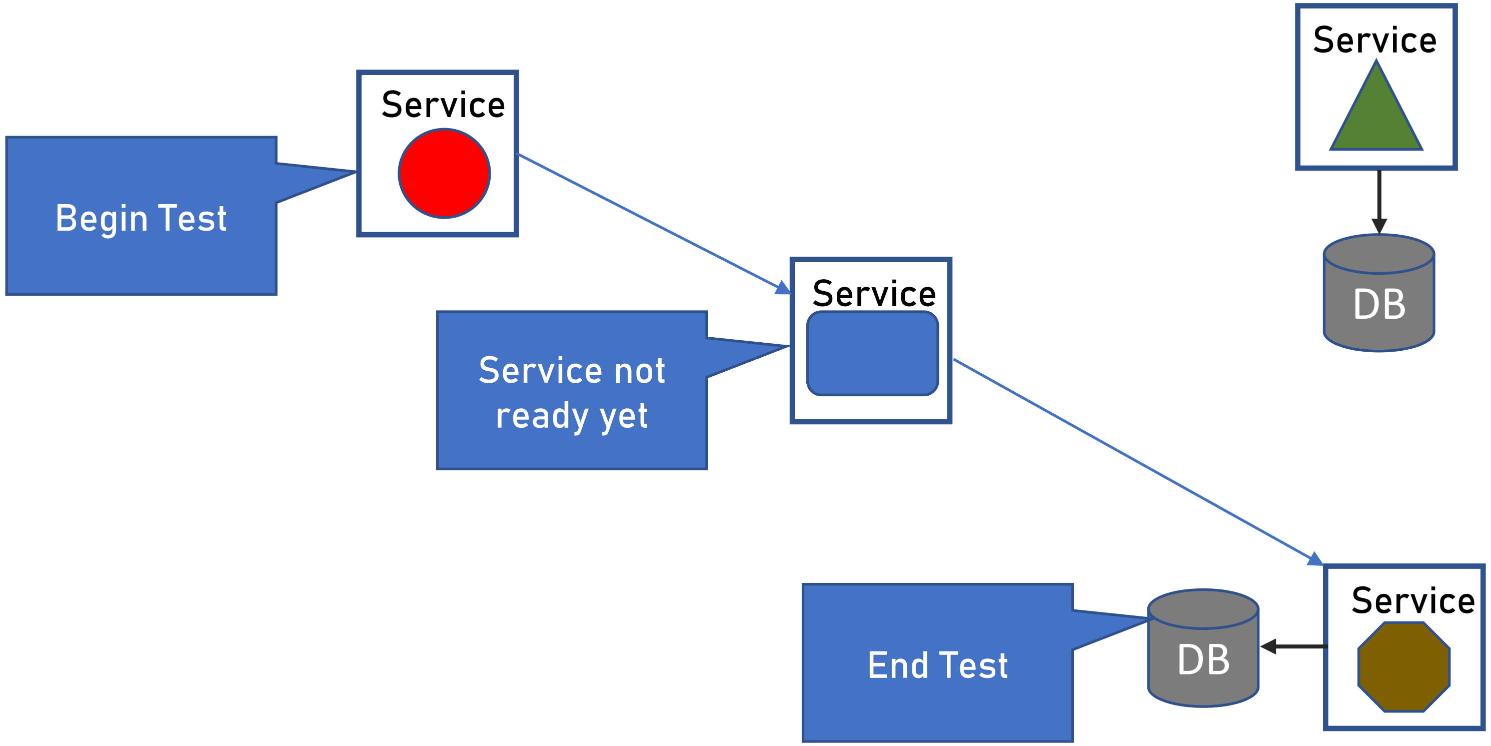
Tests Types

- Unit Tests
- Integration Tests
- End-to-End Tests

Challenges with Microservices Testing

- Microservices systems have a lot of moving parts

- Testing and tracing all the services is not trivial



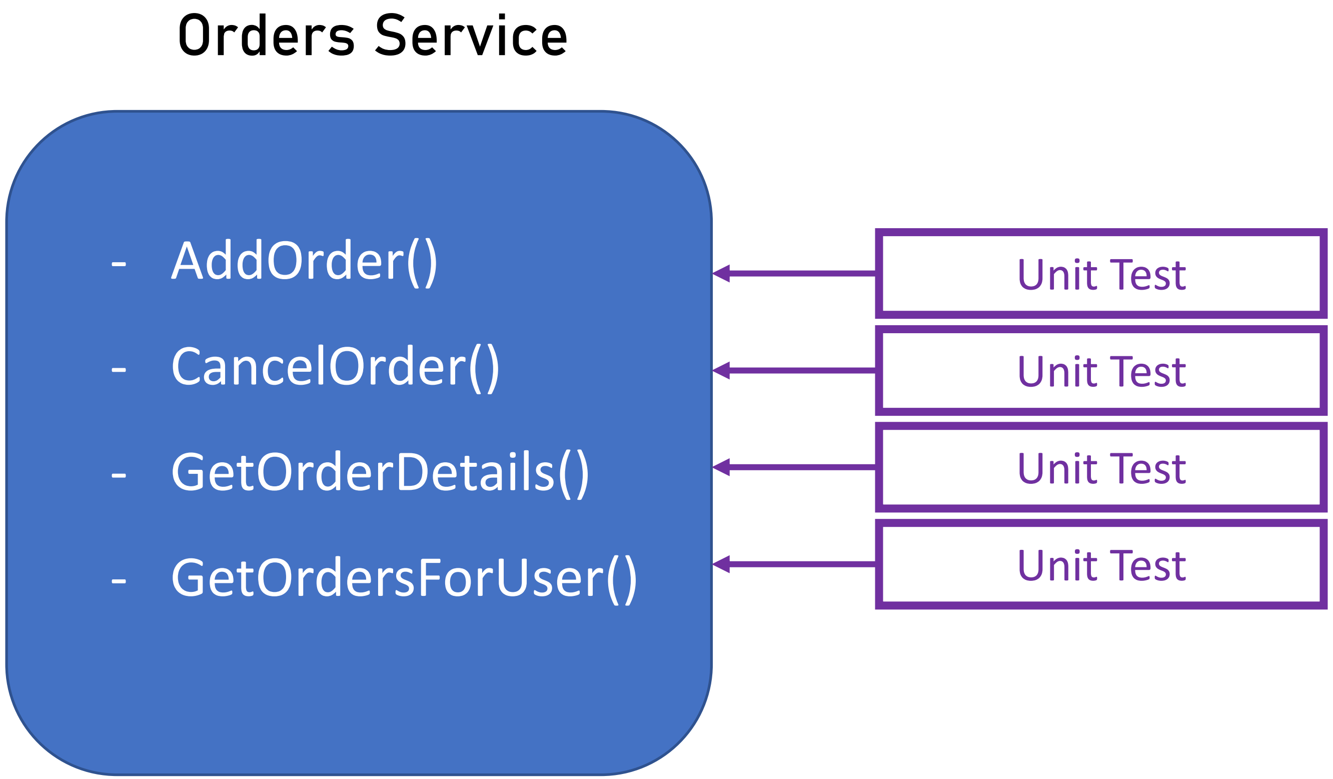
Challenges with Microservices Testing

- Main challenges:
  - Testing state across services
  - Non-functional dependent services

Unit Tests

- Tests individual code units
  - Method, interface, etc.
- In-process only
- Usually automated
- Developed by the developers

Unit Tests



Orders Service

- AddOrder()
- CancelOrder()
- GetOrderDetails()
- GetOrdersForUser()

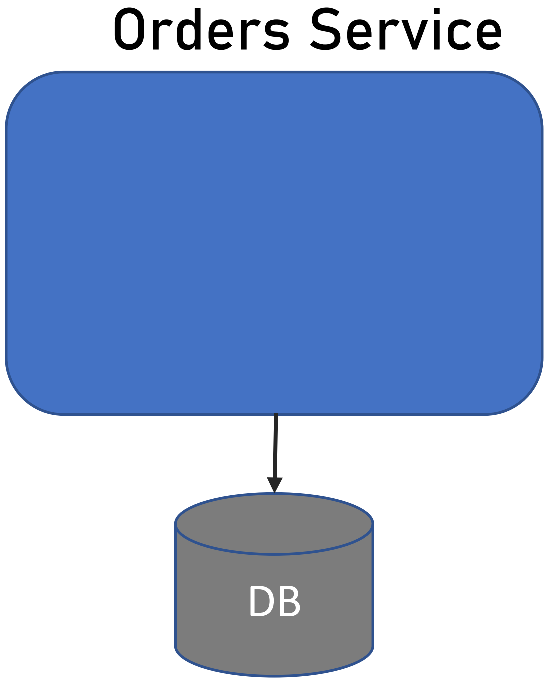
Unit Tests in Microservices

- No different
- Test only in-process code
- Use the same frameworks and methodologies

Integration Tests

- Test the service’s functionality
- Cover (almost) all code paths in the service
- Some paths might include accessing external objects
    - Database, other services

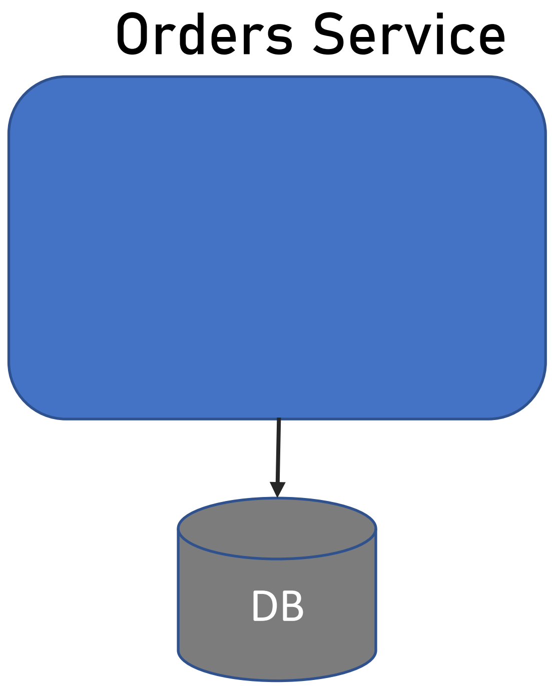
Integration Tests



What happens if the database (or other external service) is not ready?

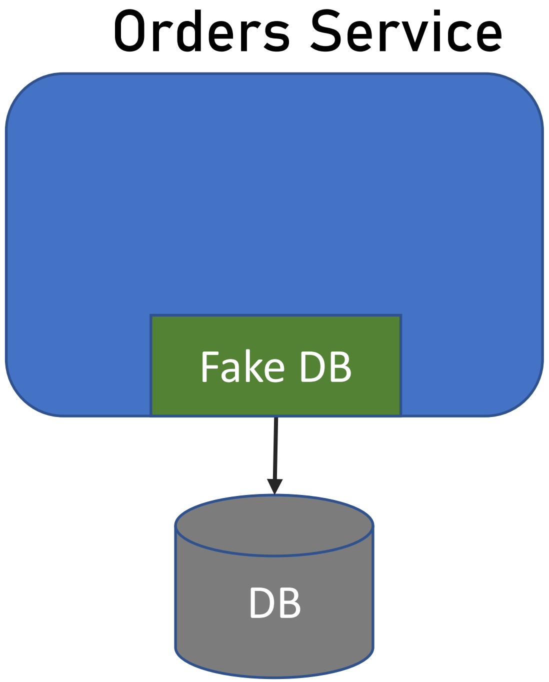
Test Double

- Pretends to be the real object / service to allow testing
- Three types:
  - Fake
  - Stub
  - Mock



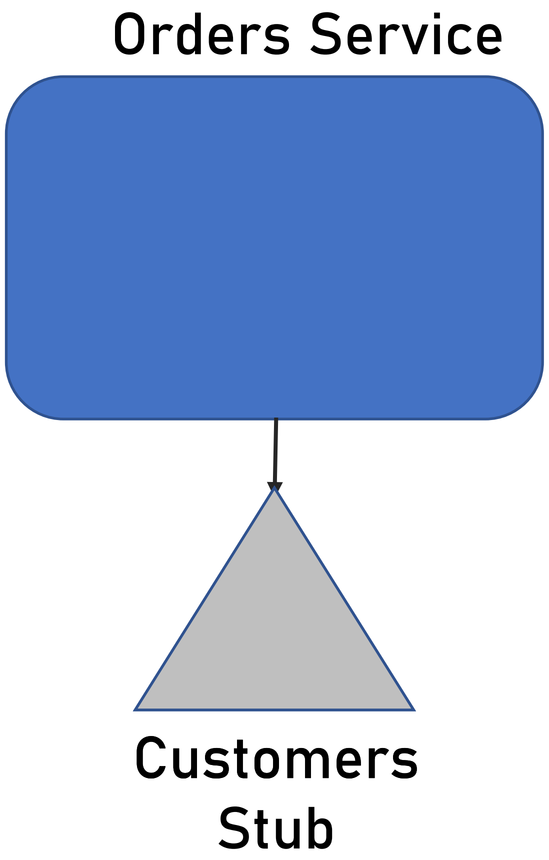
Fake

- Implements a shortcut to the external service
- For example – stores data in-memory
- Many times implemented in-process
- Requires code change in the code



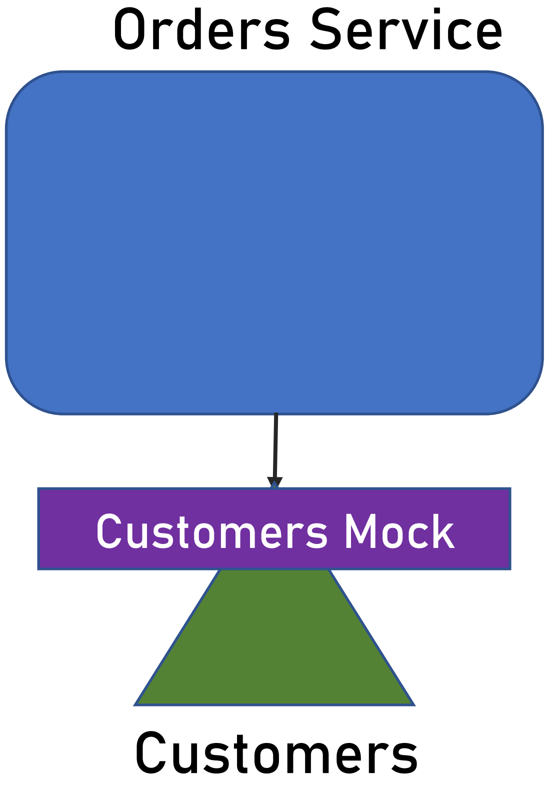
Stub

- Holds hard-coded data
- Usually replaces data stored in a DB
- Allows simulating data services quickly
- No code change required



Mock

- Verifies access was made
- Holds no data
- No code change required



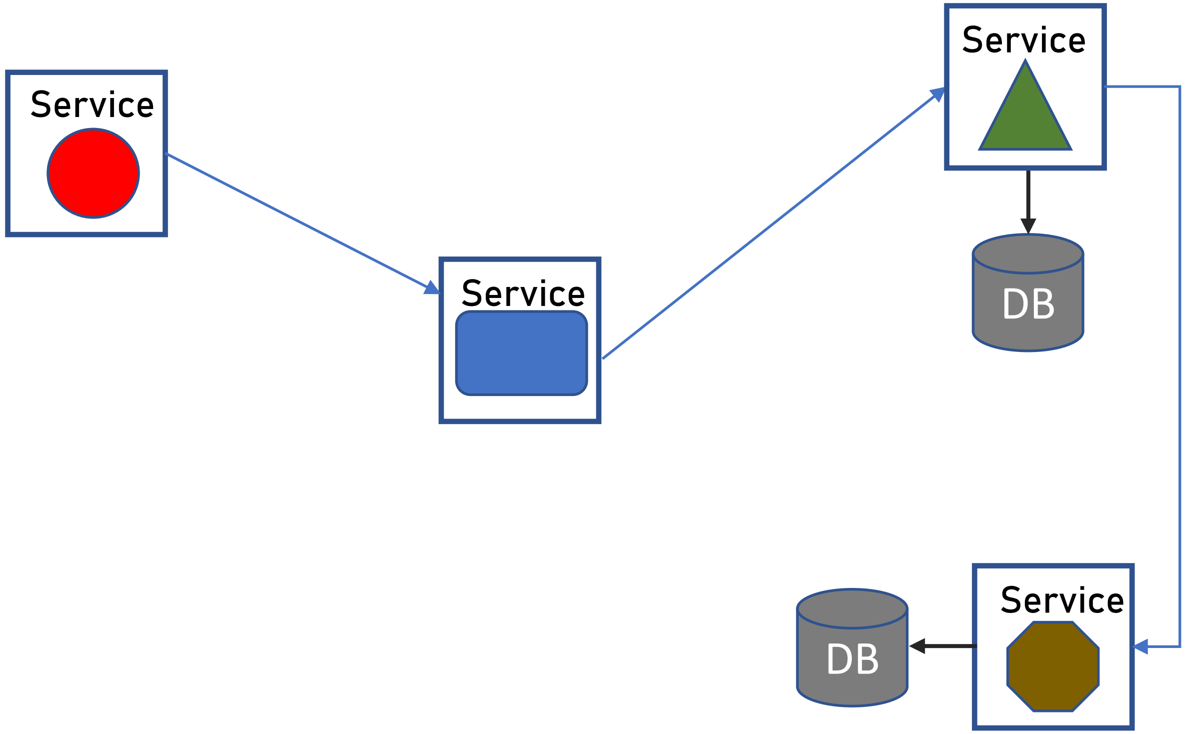
Integration Tests

- Use the service's API
- Developed and conducted by the QA team
- Should be automated
- Most unit testing frameworks support integration test

End-to-End Tests

- Test the whole flow(s) of the system
- Touch all services
- Test for end state

End-to-End Tests



End-to-End Tests

- Extremely fragile
- Require code
- Usually used for main scenarios only

Testing Microservices - Summary

- Extremely important
- Focus on the integration tests
- As an Architect:
    - make sure there is a test automation framework in place
    - Be involved in the test results analysis, may require architecture changes