Introduction to Microservices

Lotfi ben Othmane

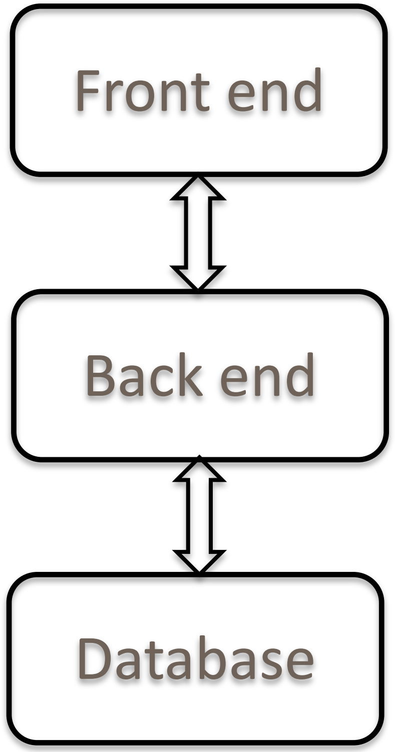
Monolithic Applications

Front end

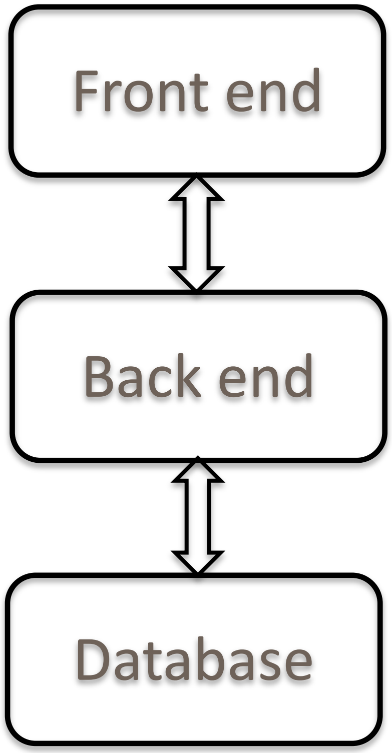
- Send username
- Send password
- Send commands
- Set application session
- Display command results
- Request logout

Back end

- Accept username
- Verify password
- Create application session
- Execute commands
- Close session



Monolithic Applications

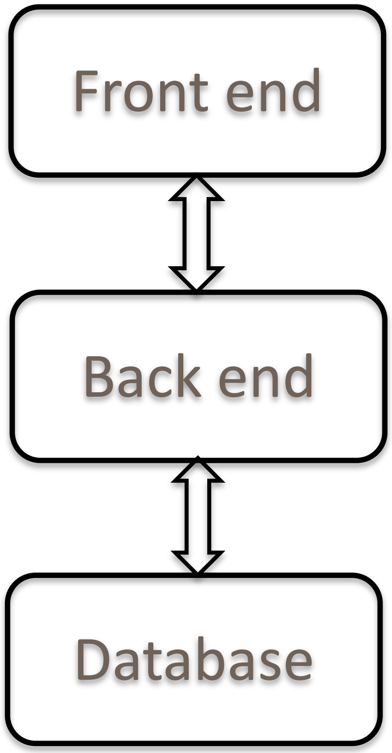


The code may be organized into modules

1. Dependency problem - Adding new features and even bug fixes requires changes to many components and redeployment of all the application

2. Interoperability problem - Organization is based on technology. Different teams work with different technologies

Monolithic Applications – Cont.



The code may be organized into modules

3. Scalability problem – Should apply to all the given application

4. Resilience problem – Fail affects all the application

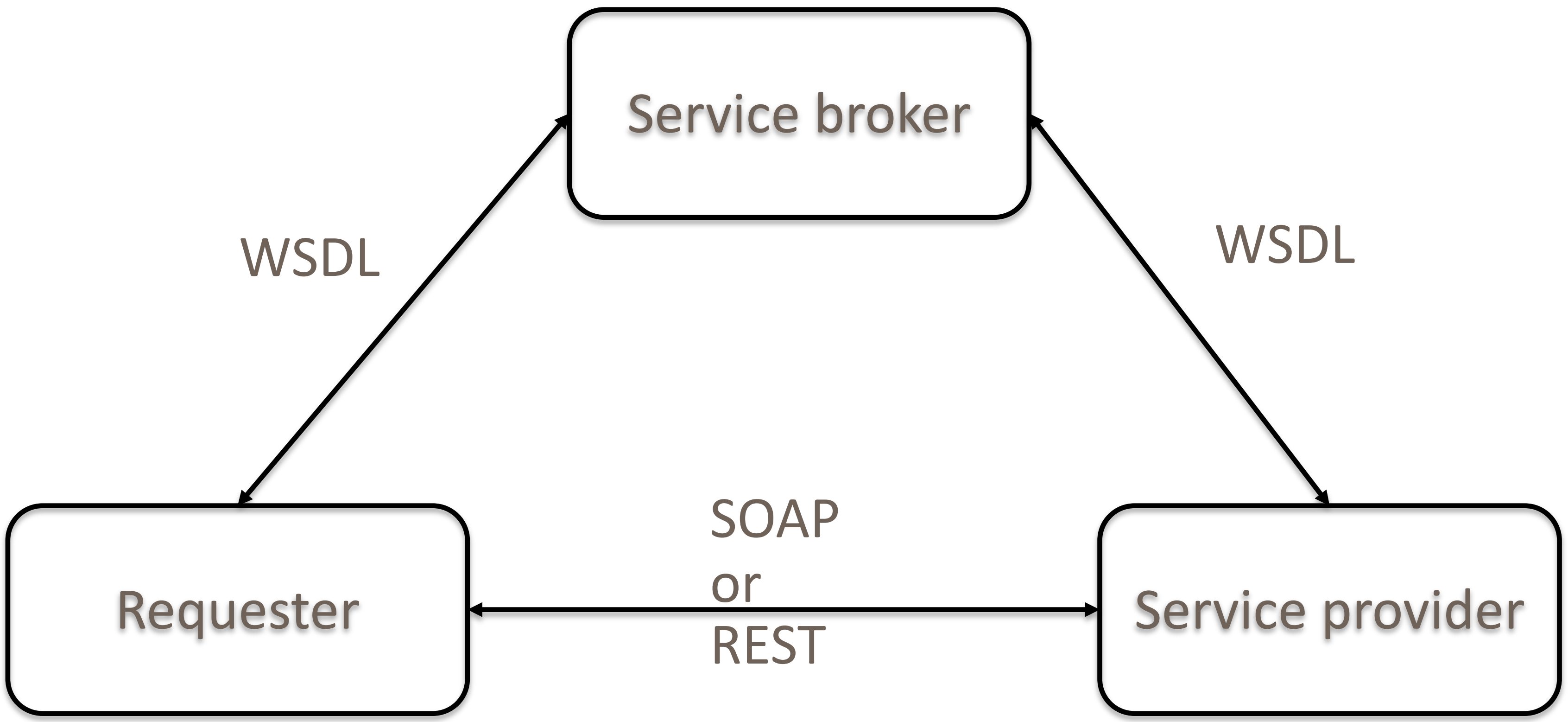
5. Consistency problem – Shared data needs to be consistent using transactions management protocol.

Monolithic Applications

There has been development of architecture styles and techniques to address the problems

- Web services for interoperability
- Transaction management with EJB
- .Net Framework
- Load balancing
- Etc.

Web Services



Micro-services

Microservices concept was first discussed in a workshop of software architects, Venice, 2011

First presentation of microservices by James Lewis:

http://2012.33degree.org/pdf/JamesLewisMicroServices.pdf

Microservice

An approach to develop a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

These services are built around business capabilities and independently deployable by fully automated deployment machinery.

https://www.martinfowler.com/articles/microservices.html

The Unix Philosophy

1. One program should fulfill only one task
2. Programs should be able to work together
3. A universal interface should be used—e.g., text stream

https://en.wikipedia.org/wiki/Unix_philosophy

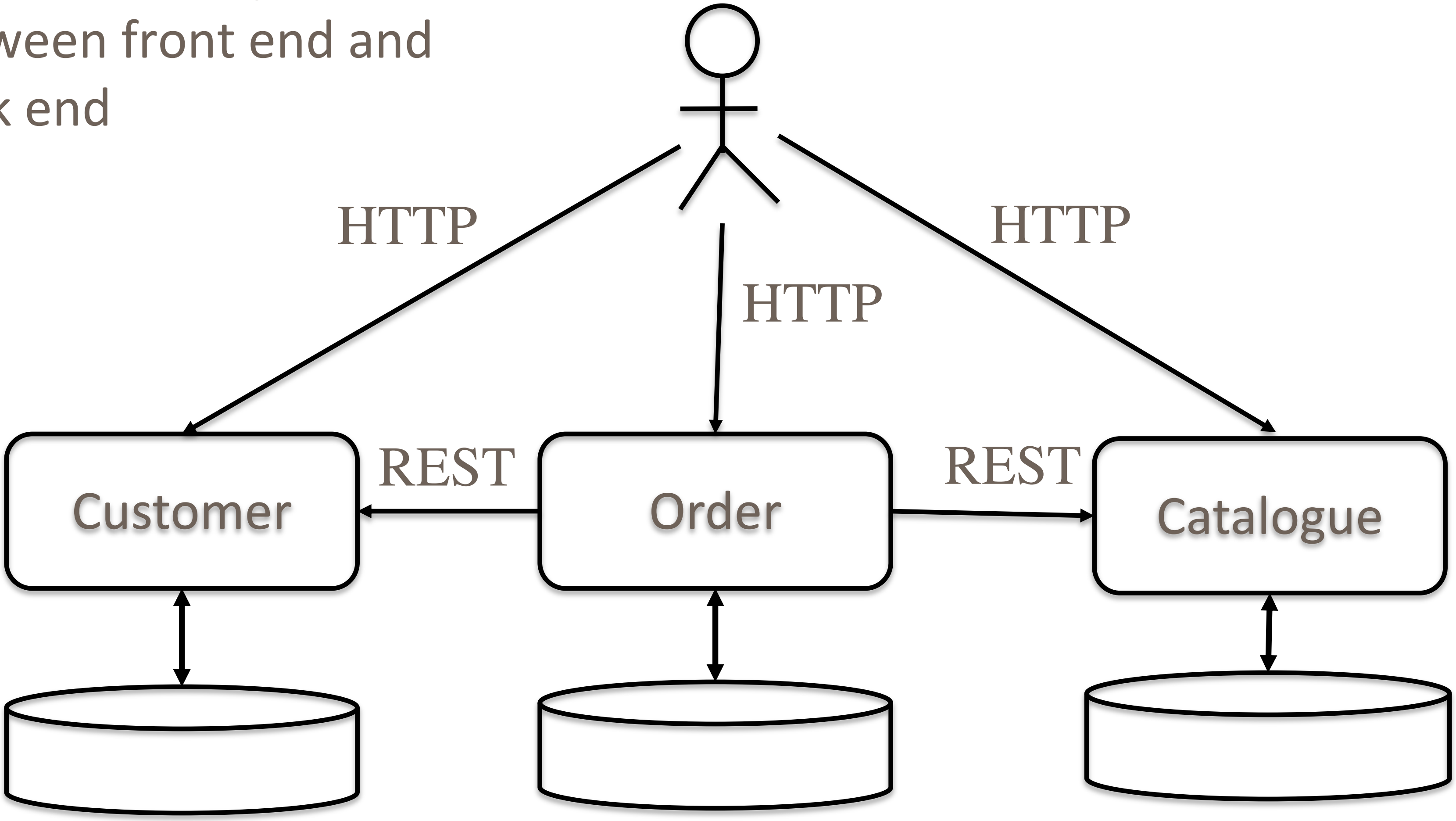
Dependency Problem

Solution

1. The boundary is based on business context not technology
   - No separation between front end and back end
2. Orchestration is implemented in microservices not in infrastructure or communication
   - Threads and workflows are managed by microservices,
3. Each microservice has a clear interface
4. Each microservice manages its own data
5. Microservices run on independent processes
   - Could be deployed independently

Dependency Problem

Solution: No separation between front end and back end

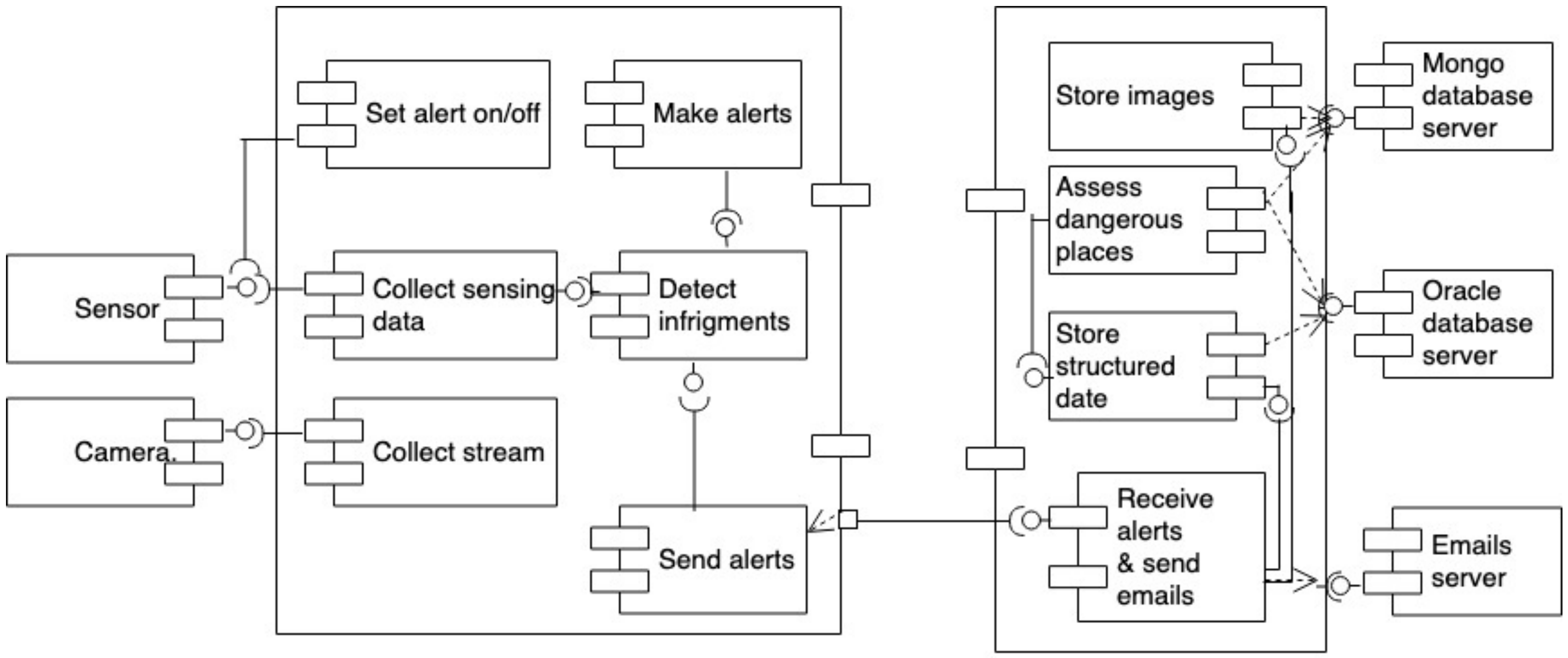


Dependency Problem

- One of the main challenges in microservice architecture is to identify cut-points
    - Identify independent components

- Principle 1: Split is based on business capabilities boundaries
- Principle 2: Future changes should require updates to one microservice—minimize propagation of changes

Dependency Problem



Criteria for Creating New Micro services

1. Introduction of different data models
2. Mixing of synchronous and asynchronous communication
3. Incorporating additional services
4. Different load scenarios for different aspects of the service

Team Coordination

How does microservice style help to coordinate a team?

Interoperability Problem

Web service solution

- Web service addresses this by allowing communication between web services using SOAP or REST

Microservice solution

- Use lightweight communication mechanisms such as REST and RPC

Consistency Problem

- Web services support transactions for consistency – May be needed in some contexts
- Web services run on one process

Consistency Problem

Microservices related-characteristics

- Asynchronous communication
- No shared data –or minimum shared
- No management of service states for consistency

Microservices solution

- Compensation operation for inconsistency
- Do not use central system for consistency

➢ Implement logic to detect inconsistency and to trigger corrective operations

Resilience Problem

Potential failure

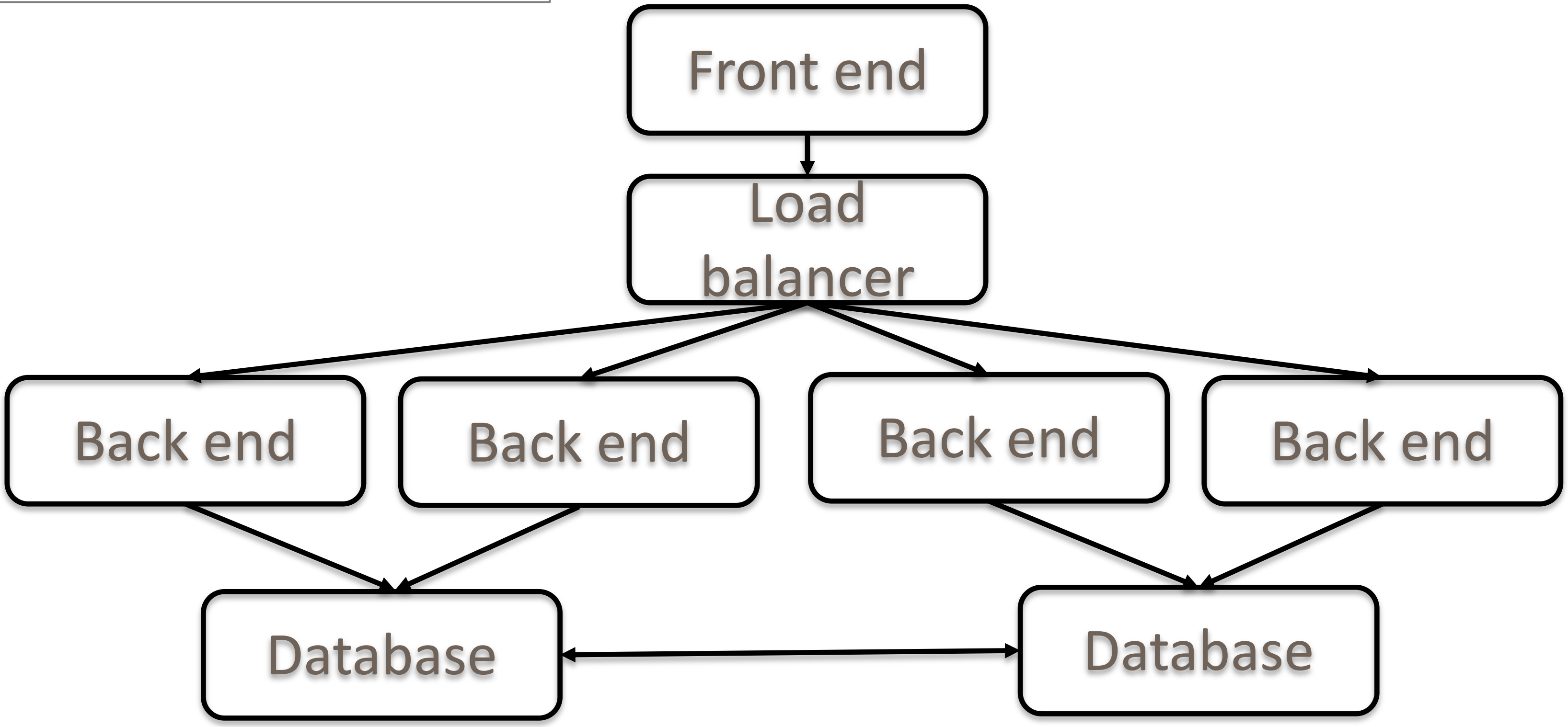
- Service might have bugs –crash
- Service may become unavailable due to hardware or network problem
- Service may become slow to respond

➢ Plan for eventual failure

- How should/must the microservice behave in the case of failure of each of the dependencies?
- Use of circuit breaker to handle failure: monitor microservices and trigger correction in case of failure

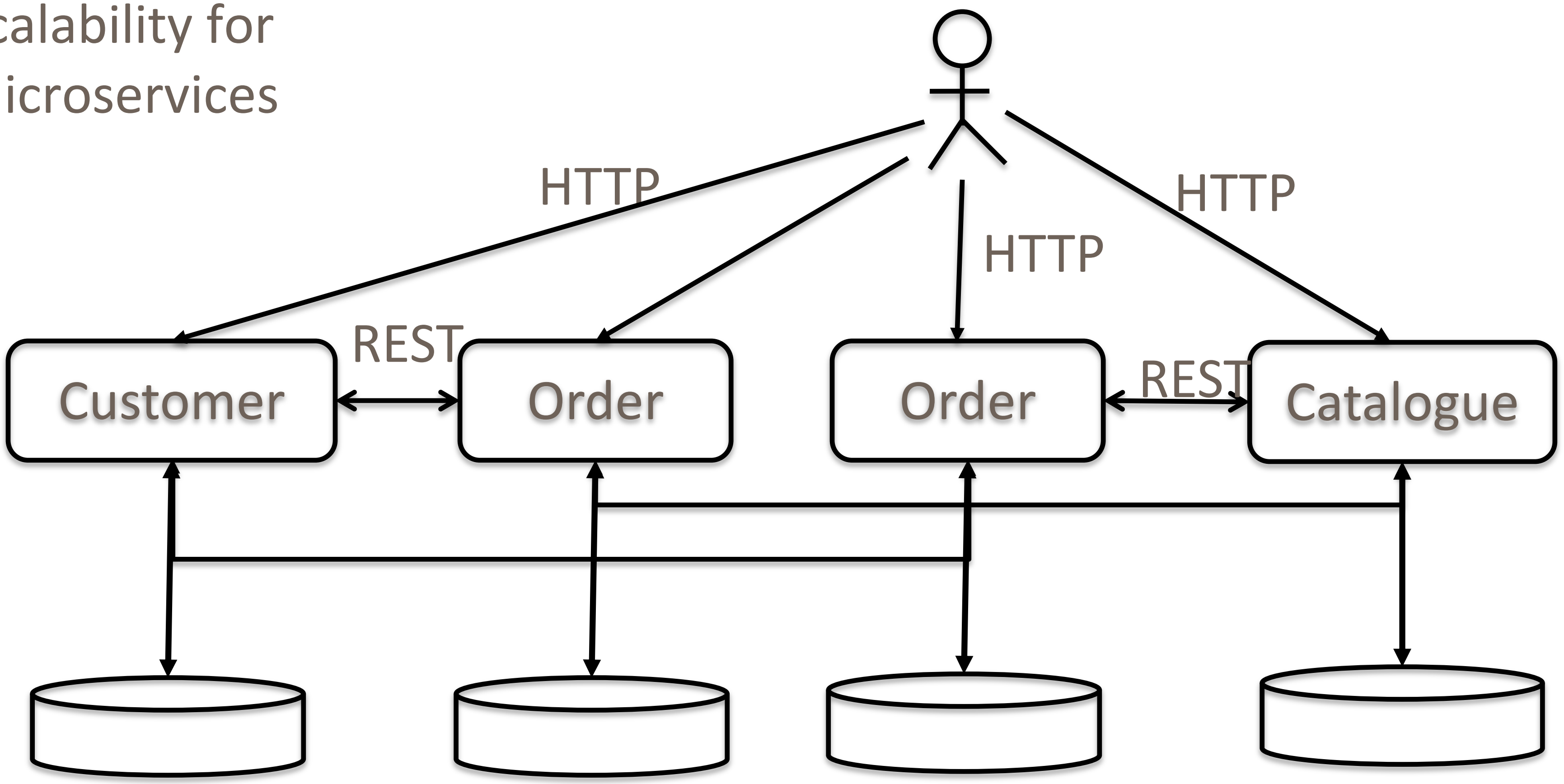
Scalability Problem

Scalability for monolithic applications



Scalability Problem

Scalability for microservices



Characteristics of Microservices

1. Organized around business capabilities
2. Products not Projects – developers support the product
3. Smart endpoints and dumb pipes – request-logic-response
4. Decentralized governance – technology choices
5. Decentralized data Management
6. Infrastructure automation – continuous development
7. Design for failure – consider failure
8. Evolutionary design – rewriting a component without affecting its collaborators

Resources

- Microservices – Flexible software architecture
  by Eberhard Wolff
  http://microservices-book.com/content.html

- Microservices -- A definition of this new architectural term
  by James Lewis and Martin Fowler
  https://www.martinfowler.com/articles/microservices.html

Summary

We’ve seen how microservices addresses the problems of:

1. Dependencies
2. Interoperability
3. Scalability
4. Resilience
5. Consistency

Thank you.