Carnegie Mellon

institute for SOFTWARE RESEARCH

DevOps-Continuous Development

Microservice Architecture

1

Carnegie Mellon University

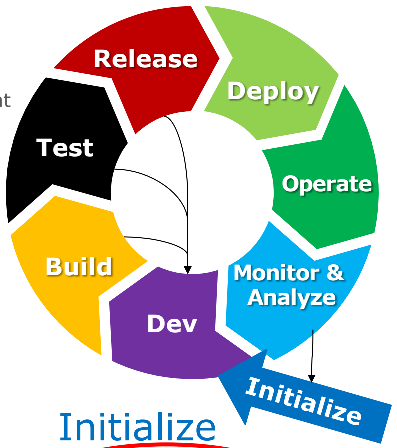
DEVOPS PROCESSES

Release
Approve for deployment

Test
Automate tests as much as possible

Build
Create an executable artifact

Dev
Perform normal development activities



Deploy
Move into production environment

Operate
Execute system and gather measurements about its operation

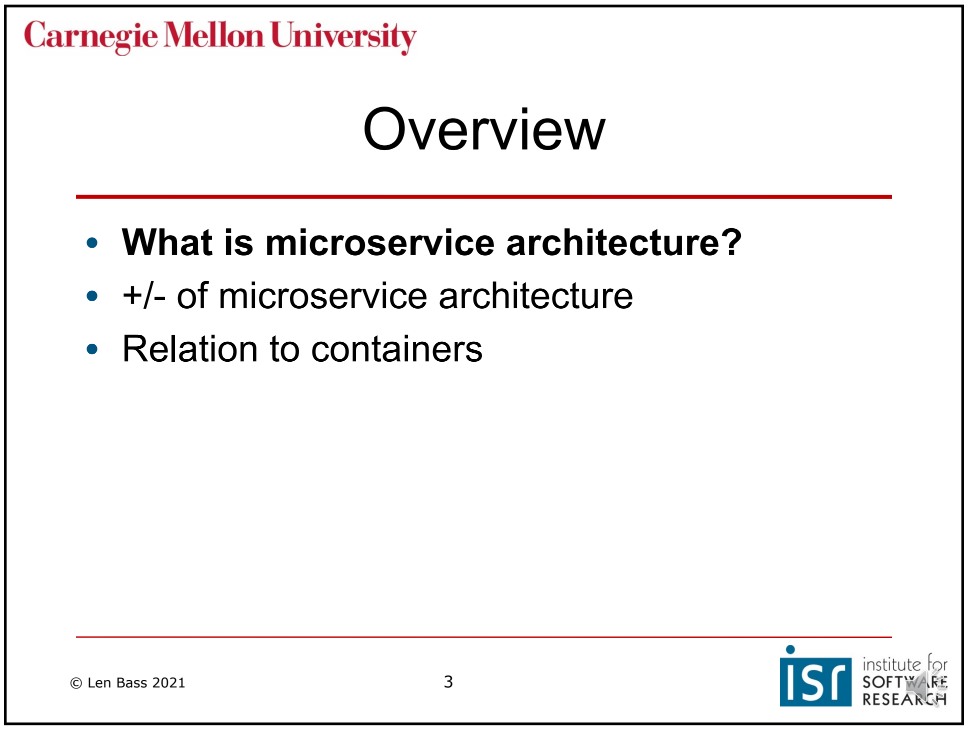
Monitor & Analyze
Display measurements taken during operation & analyze the data

Initialize
Select architecture to support other activities
Create scripts for other activities

© Len Bass 2021

institute for SOFTWARE RESEARCH

2

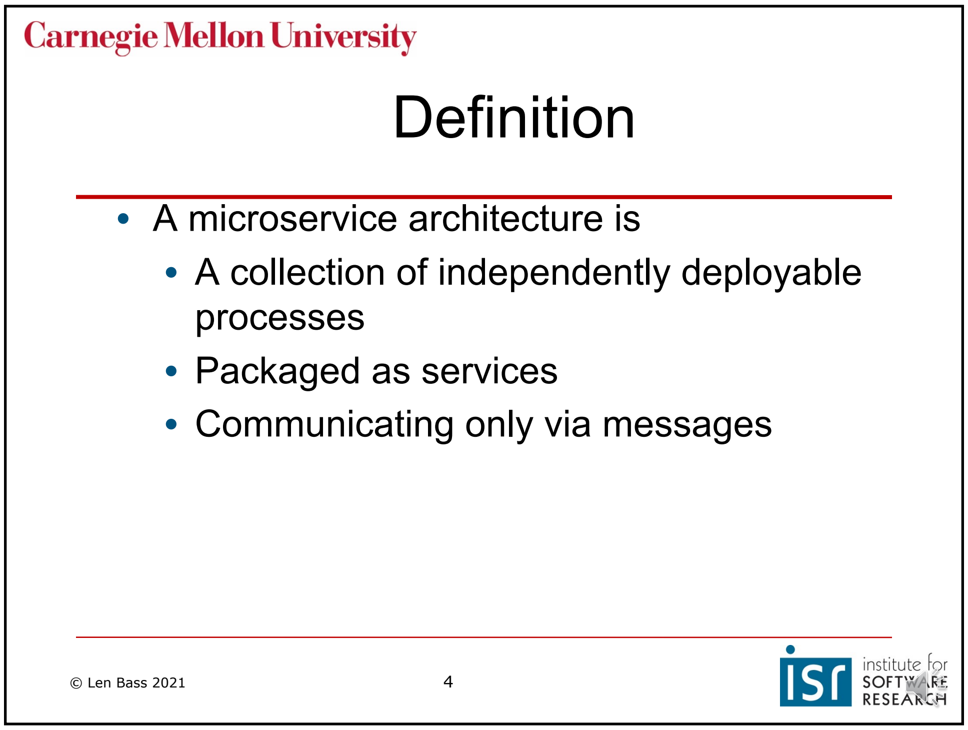

Carnegie Mellon University

Overview

- **What is microservice architecture?**
- +/- of microservice architecture
- Relation to containers

© Len Bass 2021 3

3

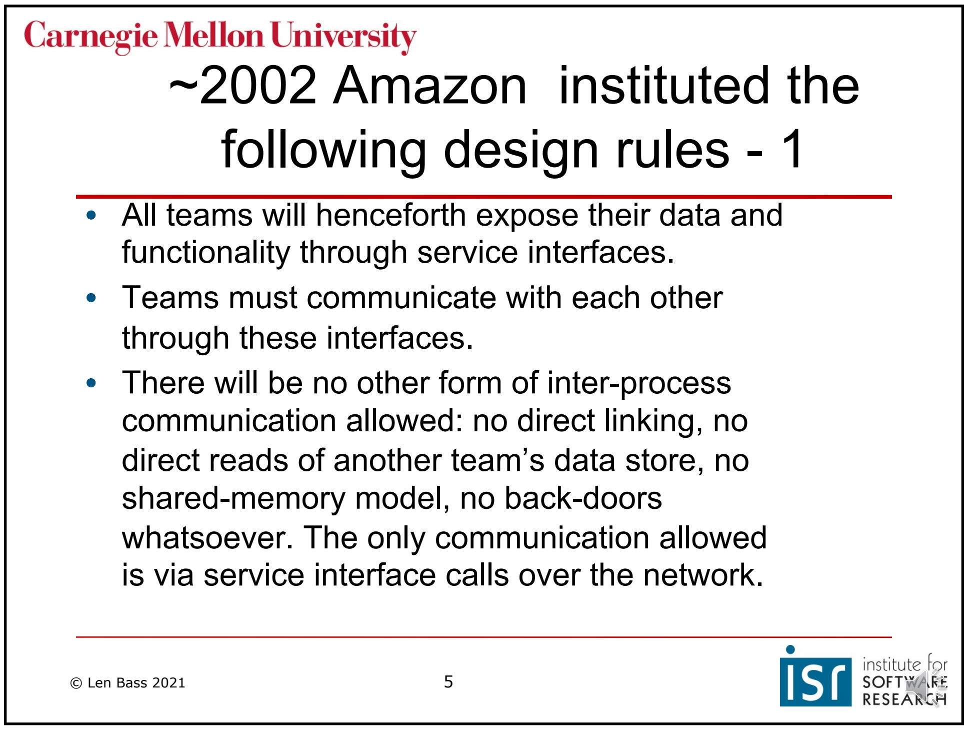

Carnegie Mellon University

Definition

- A microservice architecture is
  - A collection of independently deployable processes
  - Packaged as services
  - Communicating only via messages

© Len Bass 2021 4

4



Carnegie Mellon University

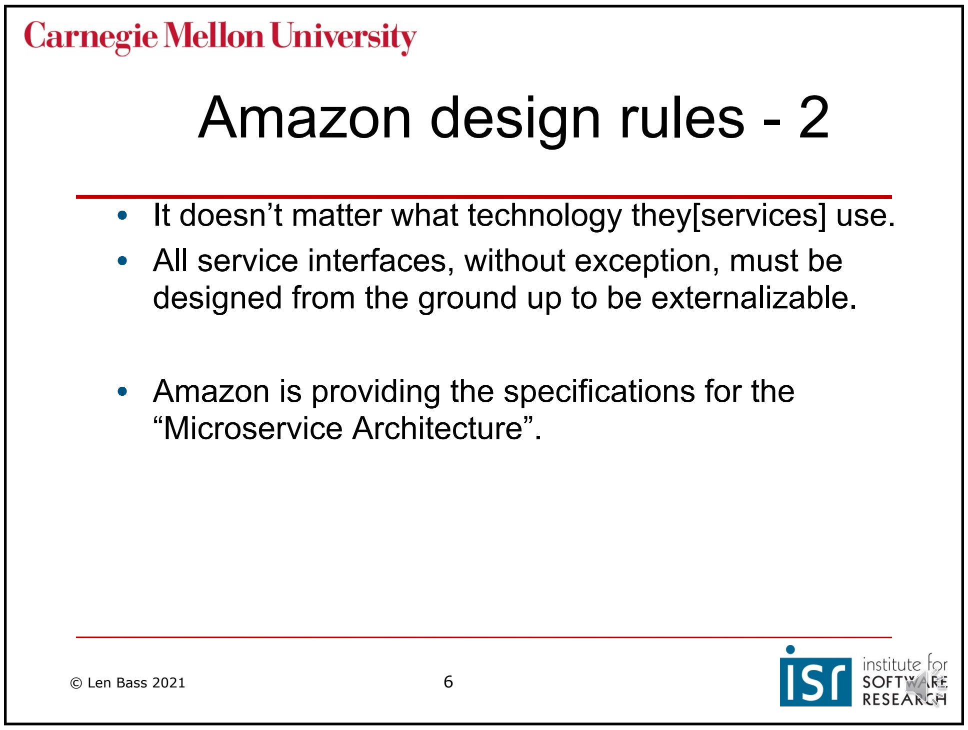
~2002 Amazon instituted the following design rules - 1

- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of inter-process communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

© Len Bass 2021 5

institute for SOFTWARE RESEARCH

5



Carnegie Mellon University

Amazon design rules - 2

- It doesn't matter what technology they[services] use.
- All service interfaces, without exception, must be designed from the ground up to be externalizable.
- Amazon is providing the specifications for the "Microservice Architecture".

© Len Bass 2021 6

institute for SOFTWARE RESEARCH

6

Carnegie Mellon University

In Addition

- Amazon has a “two pizza” rule.
- No team should be larger than can be fed with two pizzas (~7 members).
- Each (micro) service is the responsibility of one team
- This means that microservices are small and intra team bandwidth is high
- Large systems are made up of many microservices.
- There may be as many as 140 in a typical Amazon page.

© Len Bass 2021 7

institute for SOFTWARE RESEARCH

7

Carnegie Mellon University

Services can have multiple instances

- The elasticity of the cloud will adjust the number of instances of each service to reflect the workload.
- Requests are routed through a load balancer for each service
- This leads to
  - Lots of load balancers
  - Overhead for each request.

© Len Bass 2021 8

institute for SOFTWARE RESEARCH

8

Carnegie Mellon University

Digression into Service Oriented Architecture (SOA)

- The definition of microservice architecture sounds a lot like SOA.
- What is the difference?
- Amazon did not use the term “microservice architecture” when they introduced their rules. They said “this is SOA done right”

© Len Bass 2021 9

institute for SOFTWARE RESEARCH

9

Carnegie Mellon University

SOA characteristics

- Services in an SOA are typically stand alone applications.
- SOA has:
  - Enterprise service bus
  - Elaborate protocols for sending messages to services (WS*)
  - Each service may be under the control of different organization
  - Brokers
  - Etc

© Len Bass 2021 10

institute for SOFTWARE RESEARCH

10

Carnegie Mellon University

Enterprise Service Bus (ESB)

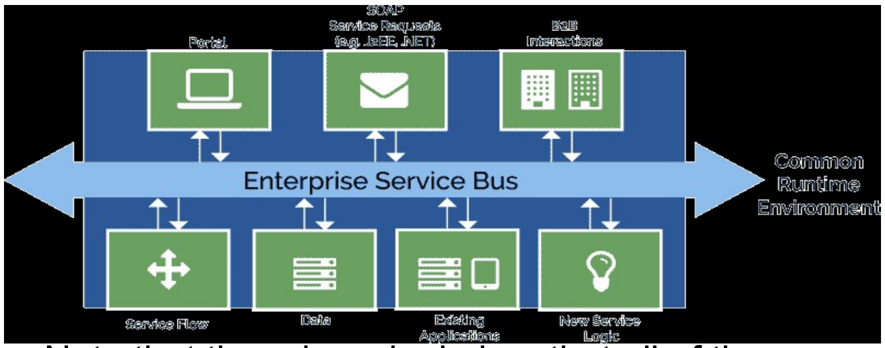
- The primary duties of an ESB are:
  - Route, monitor, and control messages between services
  - Control deployment and versioning of services
  - Provide commodity services like event handling, data transformation and mapping, message and event queuing and sequencing, security or exception handling, protocol conversion and enforcing proper quality of communication service.

© Len Bass 2021 11 institute for SOFTWARE RESEARCH

11

Carnegie Mellon University

ESB



Note that there is a single bus that all of the services interact with. Hub and spoke model

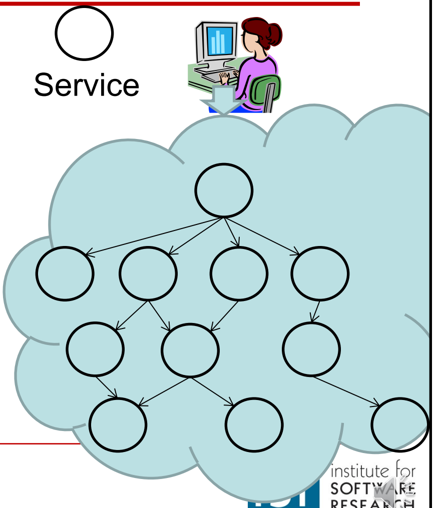
© Len Bass 2021 12 institute for SOFTWARE RESEARCH

12

Carnegie Mellon University

Micro service architecture

- Applications are collections of microservices
- Each user request is satisfied by some sequence of services.
- Most services are not externally available.
- Each service communicates with other services through service interfaces.
- Service depth may
  - Shallow (large fan out)
  - Deep (small fan out, more dependent services)



institute for SOFTWARE RESEARCH

Carnegie Mellon University

Microservice architecture supports DevOps processes

- Reduces need for coordination
- Reduces pipeline errors
- Supports continuous deployment

isr institute for SOFTWARE RESEARCH

Carnegie Mellon University

How does microservice architecture reduce requirements for coordination among teams?

- Architectural decisions can be made
  - incrementally as system evolves or
  - Initially when the system is defined..
- Microservice architecture makes most of the architectural decisions.
- Consequently they only need to be made once for a system, not once per release.

© Len Bass 2021 15

institute for SOFTWARE RESEARCH

15

Carnegie Mellon University

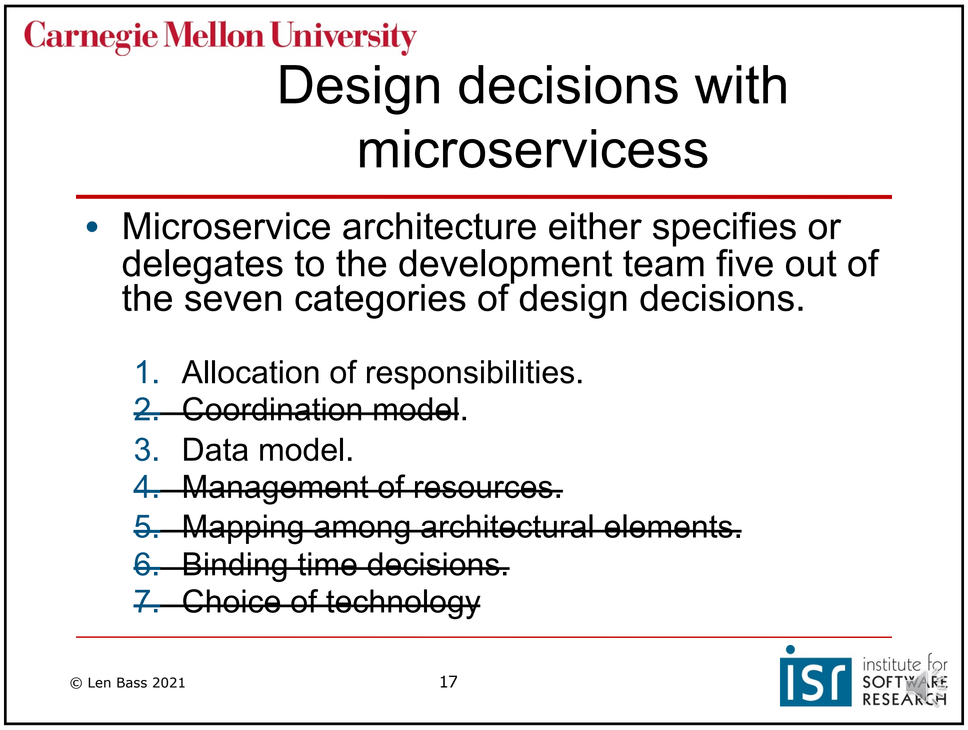
Seven Decision Categories

- Architectures can be categorized by means of seven categories
  1. Allocation of functionality
  2. Coordination model
  3. Data model
  4. Management of resources
  5. Mapping among architectural elements
  6. Binding time decisions
  7. Technology choices

© Len Bass 2021 16

institute for SOFTWARE RESEARCH

16



Carnegie Mellon University

Design decisions with microservicess

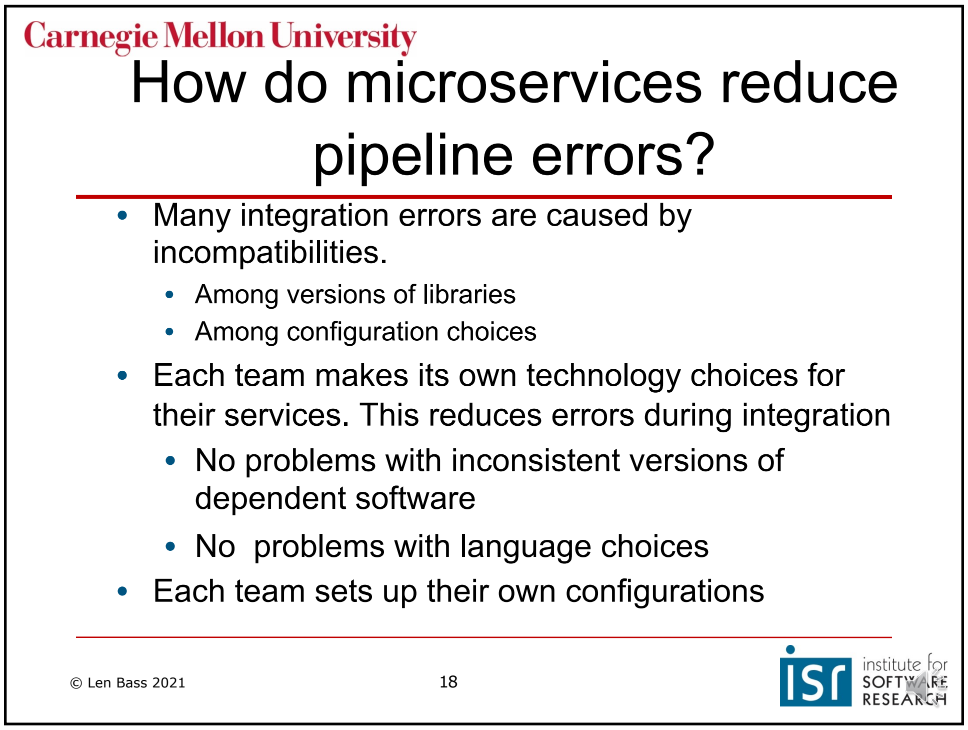
- Microservice architecture either specifies or delegates to the development team five out of the seven categories of design decisions.

1. Allocation of responsibilities.
2. ~~Coordination model~~.
3. Data model.
4. ~~Management of resources.~~
5. ~~Mapping among architectural elements.~~
6. ~~Binding time decisions.~~
7. ~~Choice of technology~~

© Len Bass 2021 17

institute for SOFTWARE RESEARCH

17



Carnegie Mellon University

How do microservices reduce pipeline errors?

- Many integration errors are caused by incompatibilities.
  - Among versions of libraries
  - Among configuration choices
- Each team makes its own technology choices for their services. This reduces errors during integration
  - No problems with inconsistent versions of dependent software
  - No problems with language choices
- Each team sets up their own configurations

© Len Bass 2021 18

institute for SOFTWARE RESEARCH

18

Carnegie Mellon University

How do microservices support continuous deployment?

- Each team can deploy their service when it is ready without consultation with other teams.
  - Since services are small, this can happen frequently
  - New feature implementation in one service can be deployed without waiting for that feature to be implemented in other services.
  - Requires architectural support – discussed in lecture on deployment

© Len Bass 2021 19

institute for SOFTWARE RESEARCH

19

Carnegie Mellon University

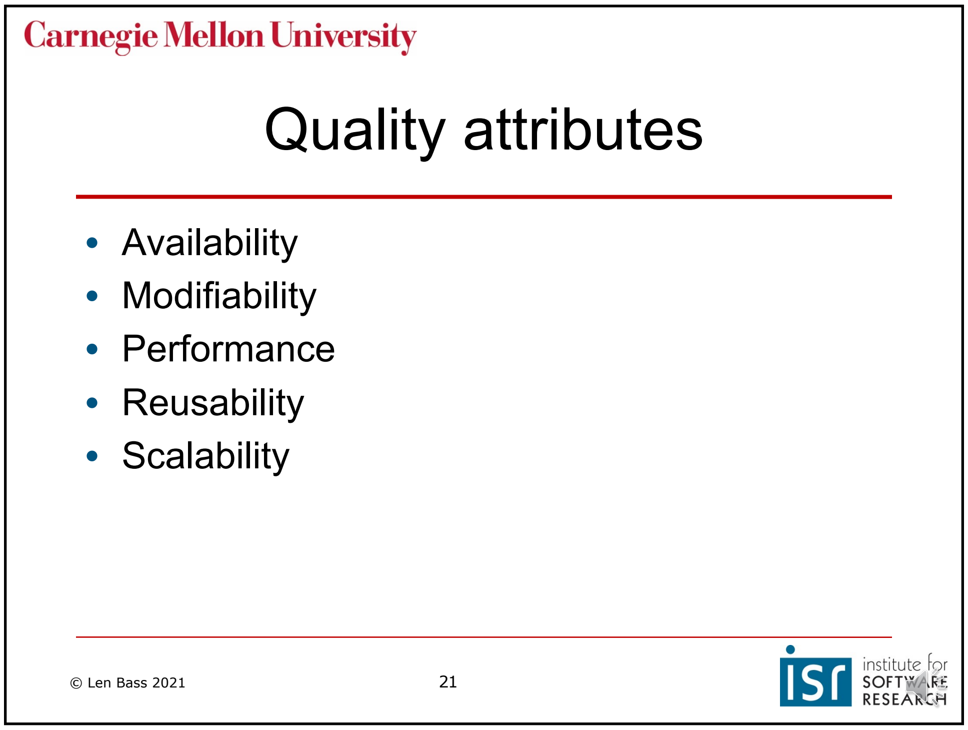
Overview

- What is microservice architecture?
- +/- of microservice architecture
- Relation to containers

© Len Bass 2021 20

institute for SOFTWARE RESEARCH

20


Carnegie Mellon University

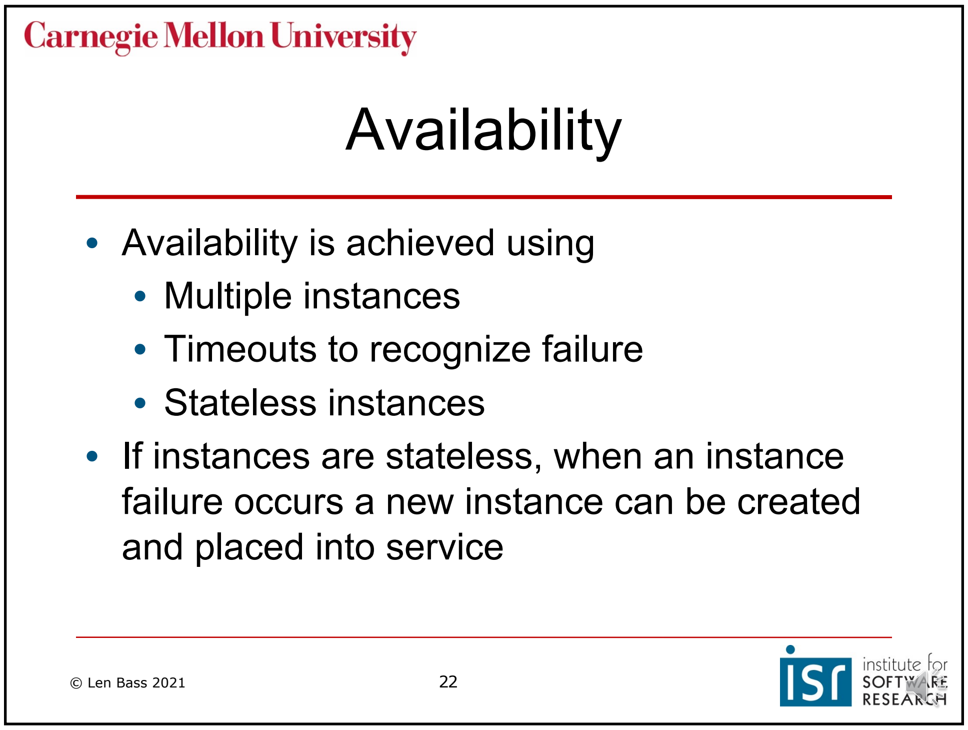
Quality attributes

- Availability
- Modifiability
- Performance
- Reusability
- Scalability

© Len Bass 2021 21

institute for SOFTWARE RESEARCH

21


Carnegie Mellon University

Availability

- Availability is achieved using
  - Multiple instances
  - Timeouts to recognize failure
  - Stateless instances
- If instances are stateless, when an instance failure occurs a new instance can be created and placed into service

© Len Bass 2021 22

institute for SOFTWARE RESEARCH

22

Carnegie Mellon University

Modifiability

- Modifiability depends on the design of the various microservices.
- Low coupling and high cohesion are design principles to achieve modifiability.
- Also, a catalog of microservices must be maintained since there will be many microservices and developers must be able to understand how functionality is distributed among them.

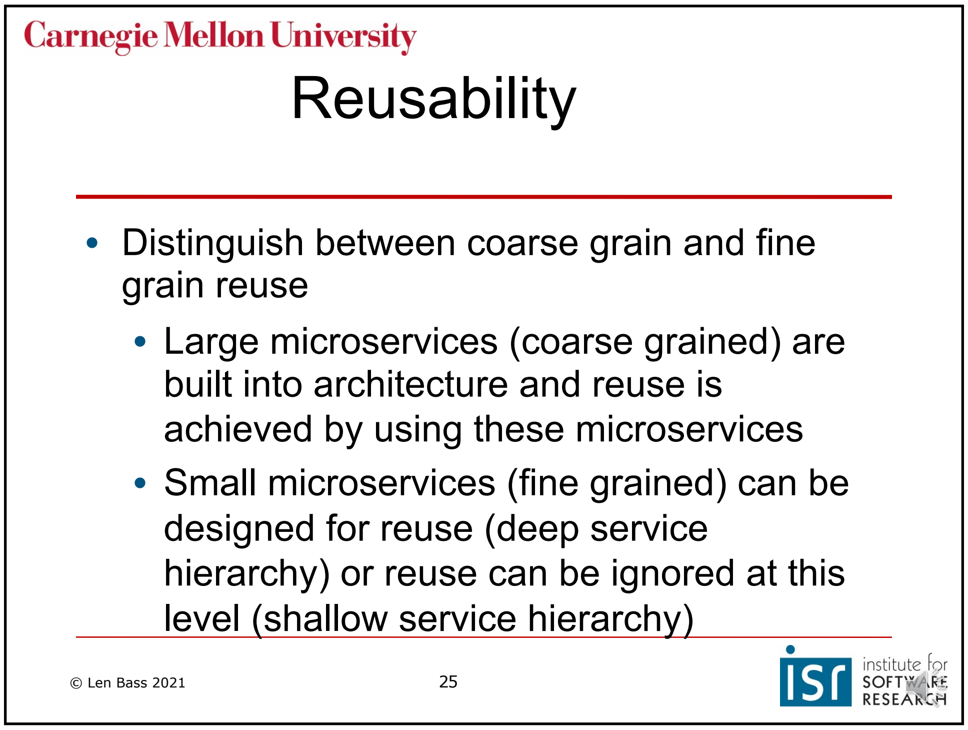
institute for SOFTWARE RESEARCH

Carnegie Mellon University

Performance

- Microservices are based on message passing.
- Network communication takes time.
- Important to measure and monitor microservices to determine where time is spent so that performance objectives can be met.
- Pods are means for improving performance

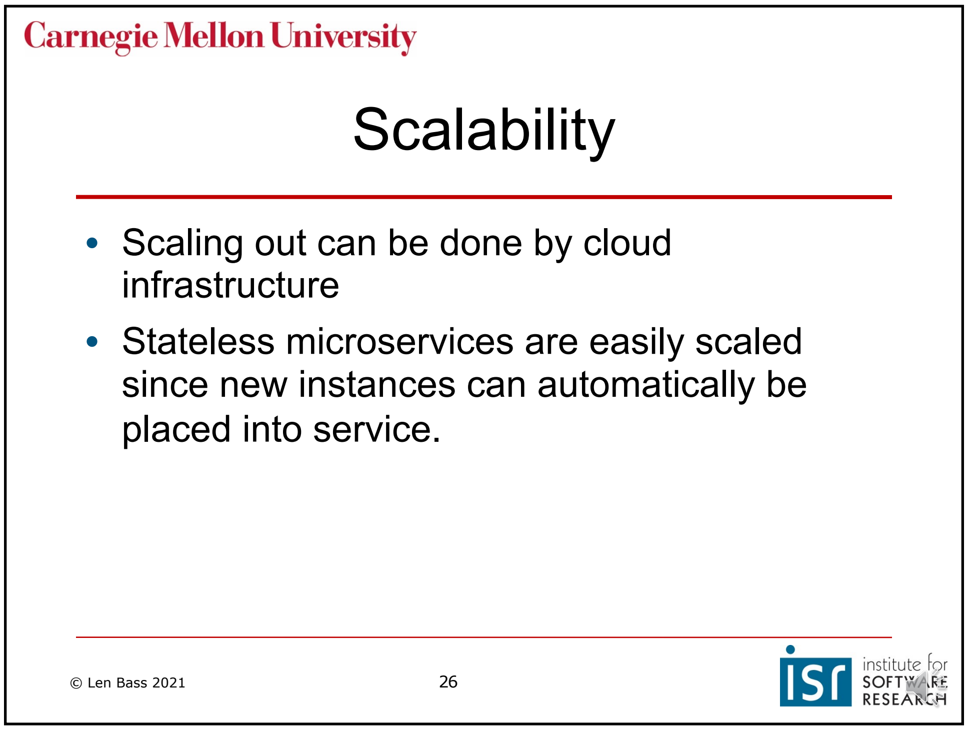
institute for SOFTWARE RESEARCH


Carnegie Mellon University

Reusability

- Distinguish between coarse grain and fine grain reuse
  - Large microservices (coarse grained) are built into architecture and reuse is achieved by using these microservices
  - Small microservices (fine grained) can be designed for reuse (deep service hierarchy) or reuse can be ignored at this level (shallow service hierarchy)

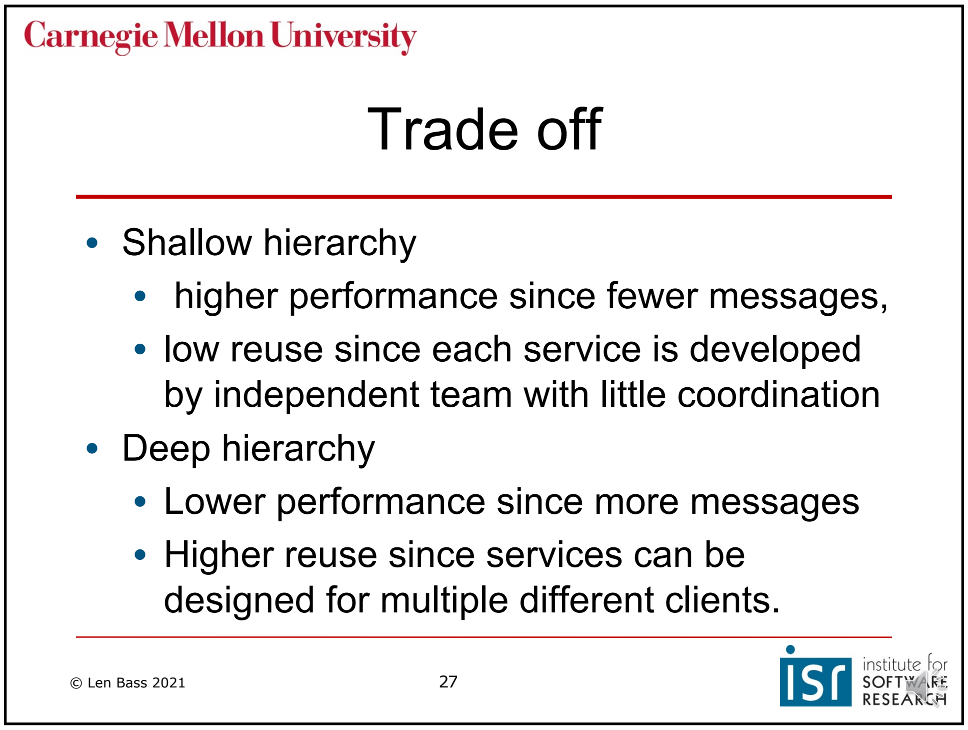
isr institute for SOFTWARE RESEARCH


Carnegie Mellon University

Scalability

- Scaling out can be done by cloud infrastructure
- Stateless microservices are easily scaled since new instances can automatically be placed into service.

isr institute for SOFTWARE RESEARCH


Carnegie Mellon University

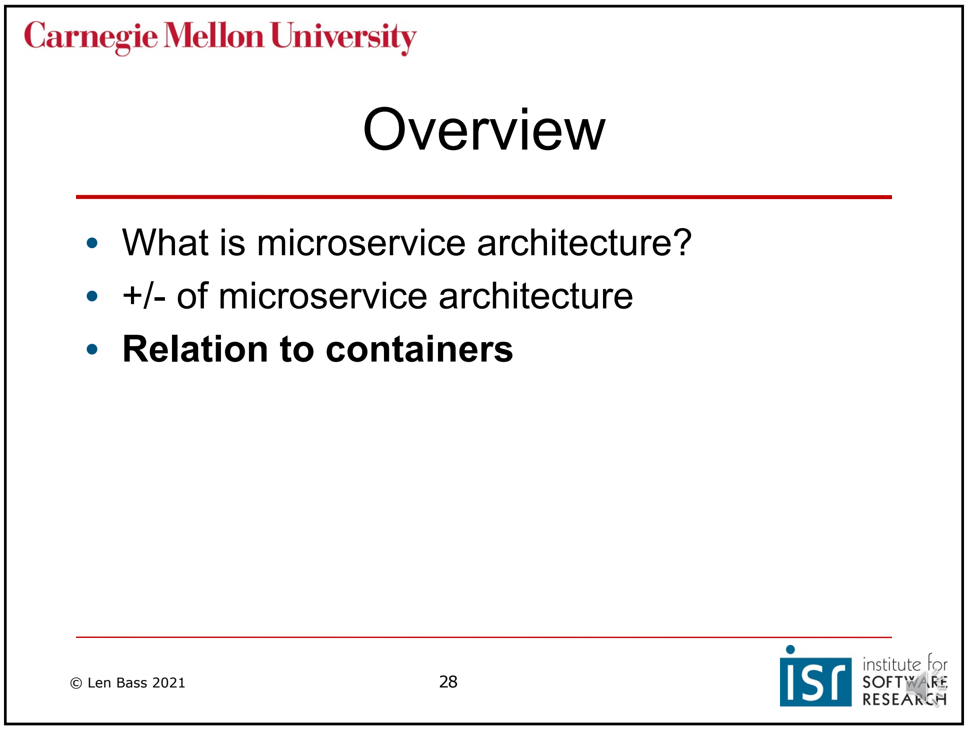
Trade off

- Shallow hierarchy
  - higher performance since fewer messages,
  - low reuse since each service is developed by independent team with little coordination
- Deep hierarchy
  - Lower performance since more messages
  - Higher reuse since services can be designed for multiple different clients.

© Len Bass 2021 27

institute for SOFTWARE RESEARCH

27


Carnegie Mellon University

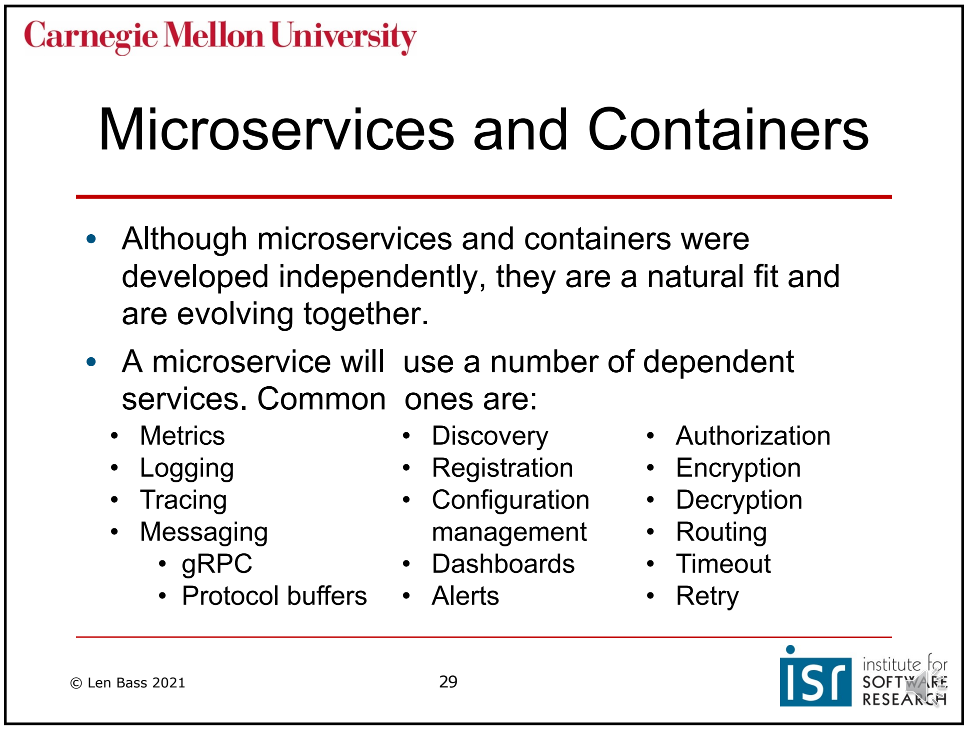
Overview

- What is microservice architecture?
- +/- of microservice architecture
- **Relation to containers**

© Len Bass 2021 28

institute for SOFTWARE RESEARCH

28



Carnegie Mellon University

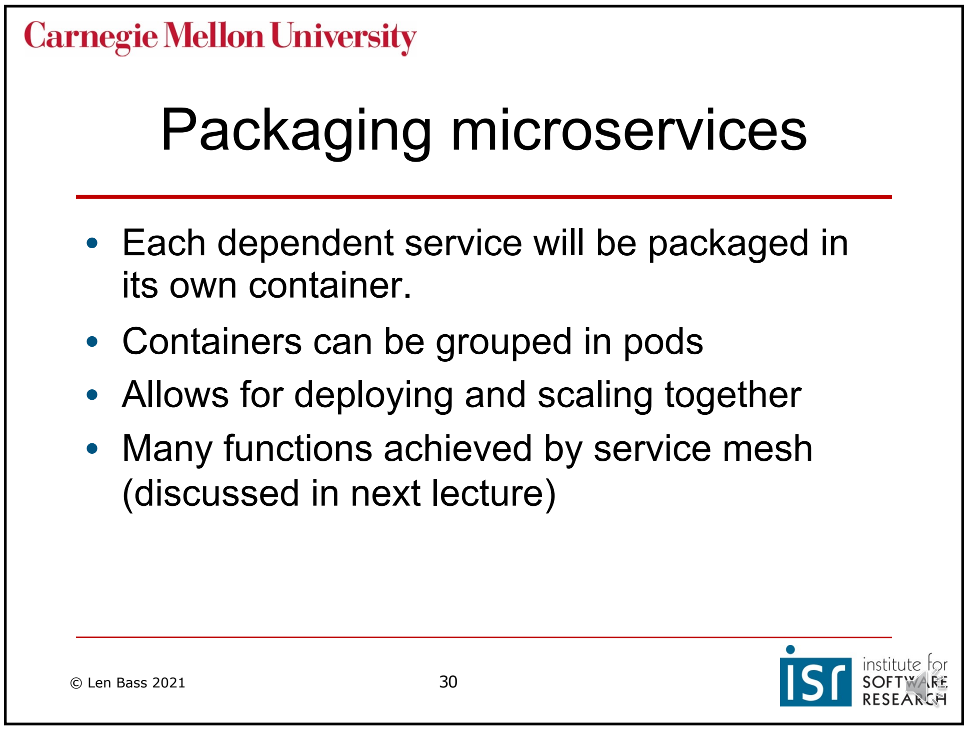
Microservices and Containers

- Although microservices and containers were developed independently, they are a natural fit and are evolving together.
- A microservice will use a number of dependent services. Common ones are:
  - Metrics
  - Logging
  - Tracing
  - Messaging
    - gRPC
    - Protocol buffers
  - Discovery
  - Registration
  - Configuration management
  - Dashboards
  - Alerts
  - Authorization
  - Encryption
  - Decryption
  - Routing
  - Timeout
  - Retry

© Len Bass 2021 29

institute for SOFTWARE RESEARCH

29



Carnegie Mellon University

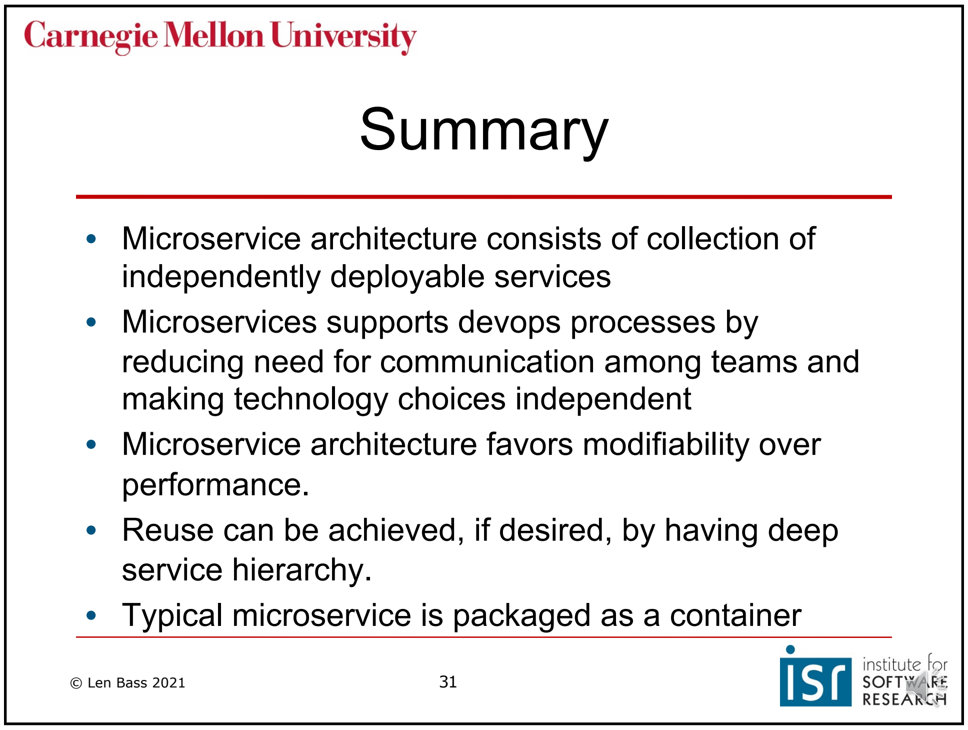
Packaging microservices

- Each dependent service will be packaged in its own container.
- Containers can be grouped in pods
- Allows for deploying and scaling together
- Many functions achieved by service mesh (discussed in next lecture)

© Len Bass 2021 30

institute for SOFTWARE RESEARCH

30


Carnegie Mellon University

Summary

- Microservice architecture consists of collection of independently deployable services
- Microservices supports devops processes by reducing need for communication among teams and making technology choices independent
- Microservice architecture favors modifiability over performance.
- Reuse can be achieved, if desired, by having deep service hierarchy.
- Typical microservice is packaged as a container

© Len Bass 2021 31

isr institute for SOFTWARE RESEARCH

31