Carnegie Mellon University
Master of Software Engineering

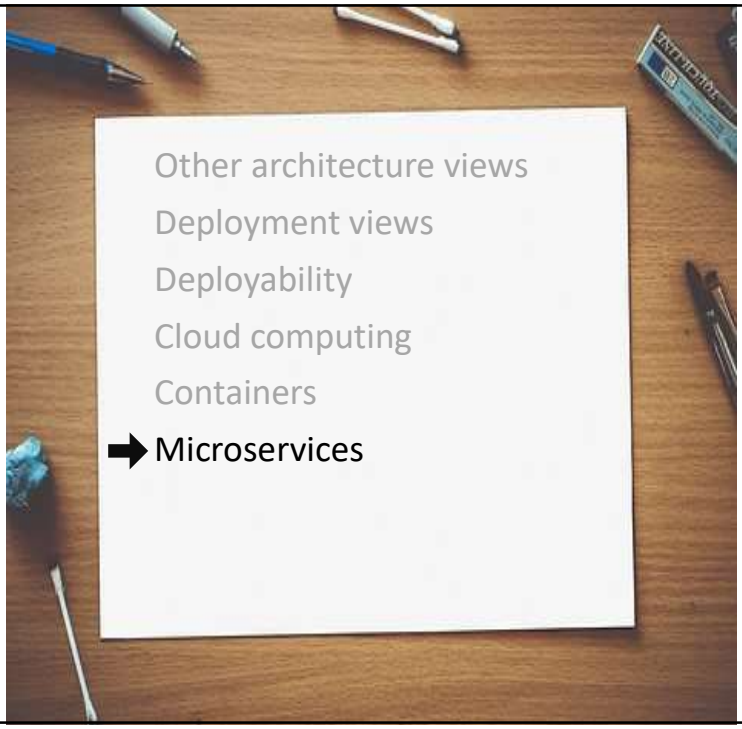
Deployment views – microservices

LG Architecture Training Program

Paulo Merson

1

Agenda



- Other architecture views
- Deployment views
- Deployability
- Cloud computing
- Containers
- ➡ Microservices

2 Carnegie Mellon University

2

Monoliths and microservices

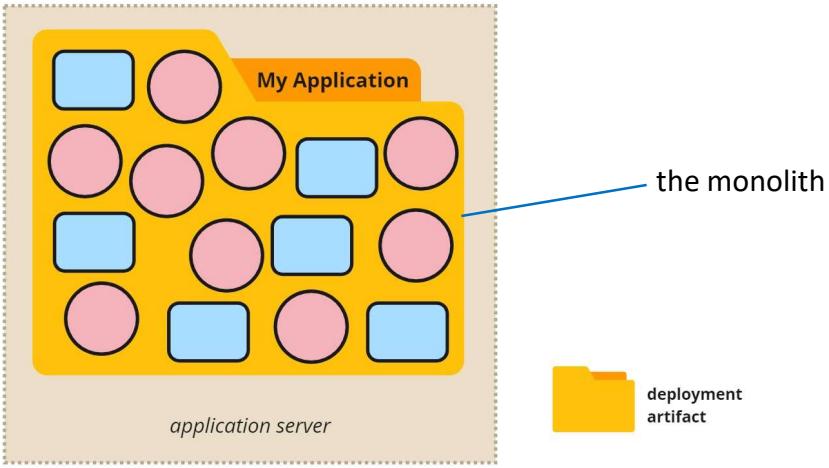
- Two general strategies for deploying traditional distributed systems on server machines:
  - Monolithic model
  - Microservices
- Microservice can be defined as an architecture pattern or style [Lewis14][Merson16][Richardson18]

Microservice and monolith are architecture styles for deployment

Monolithic style

- All services within an application are packaged together into one deployment artifact
- This deployment artifact or deployment bundle is the “monolith”
- Any small change to the application requires building and redeploying the entire monolith
- This approach was the norm for service-oriented architectures (SOA) from 2000 to 2015

Monolithic style illustrated



Microservice style

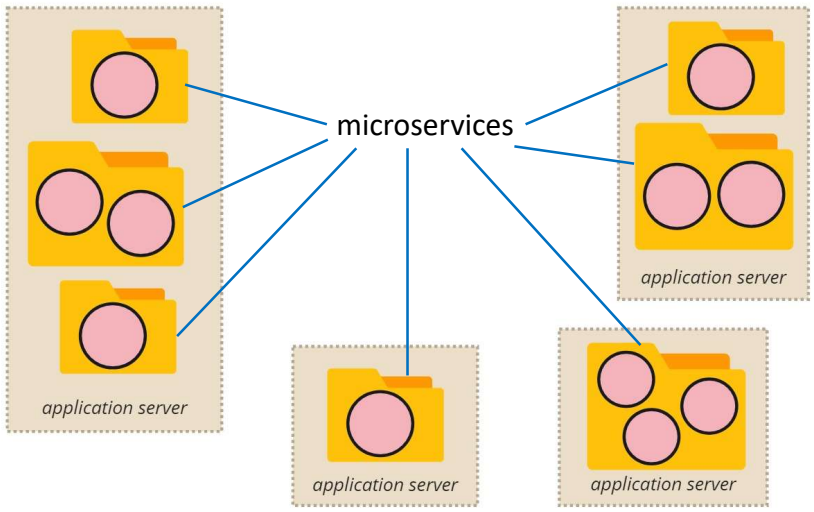
James Lewis

- Since 2014, an alternative to the monolithic style has gained space…

> *The microservice architectural style is an approach to developing 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.* [Lewis14]

> *The microservice style dictates that the deployment unit should contain only one service or just a few cohesive services. This deployment constraint is the distinguishing factor.* [Merson15a]

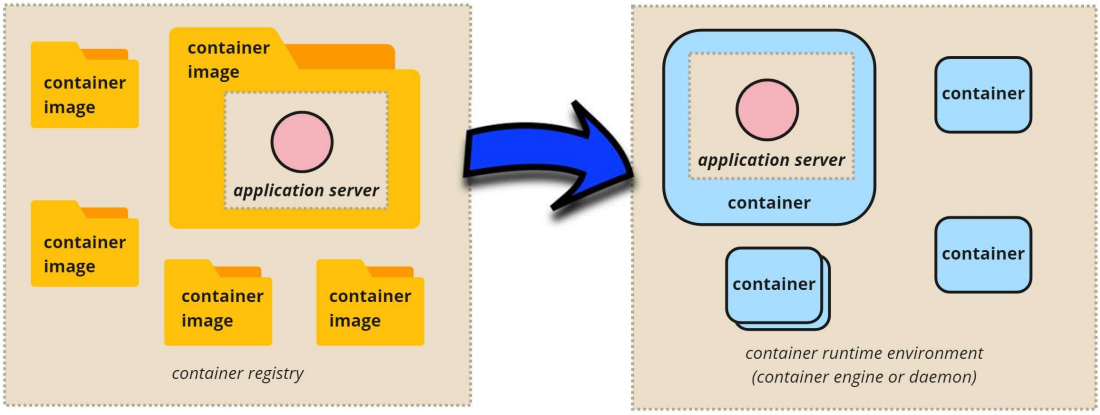
Microservice style illustrated



There are several possible variations

7

Microservices with containerization

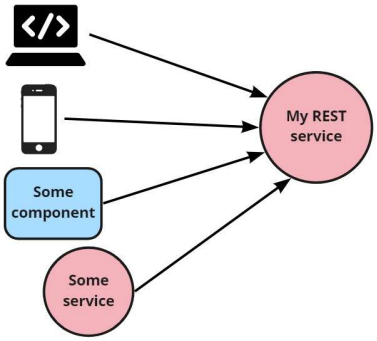


8

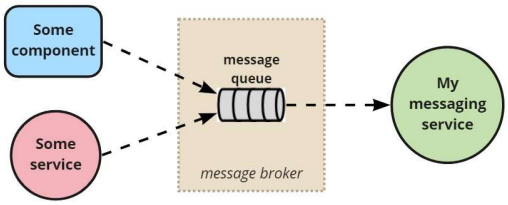
Types of connectors in microservice designs

- The microservice style does not prescribe specific technologies
- The most common types of connectors are:

Synchronous http (with REST services)



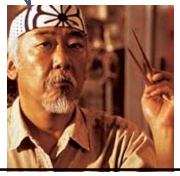
Asynchronous messages/events

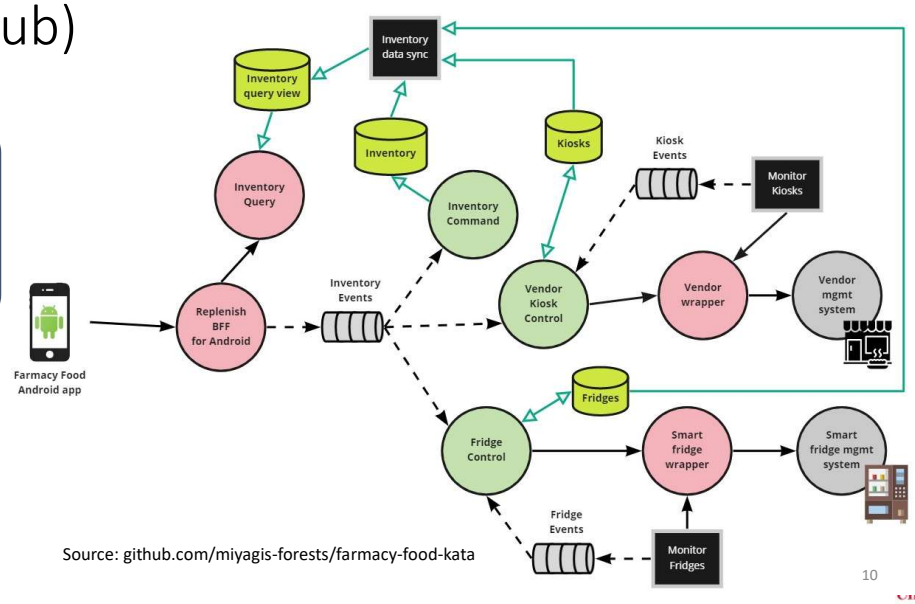


9

Example of pub-sub application (see lecture on pub-sub)

The actual name of this architecture view is Replenisher Microservice and EDA View, but where are the microservice deployment artifacts?





Source: github.com/miyagis-forests/farmacy-food-kata

10

10

Microservices benefits* (1)

- **Deployability**
  - More agility to roll out new versions of a service due to shorter build + test + deploy cycles
  - Flexibility to employ service-specific security, replication, persistence, and monitoring configurations
- **Reliability**
  - A fault affects that microservice alone and its consumers, whereas in the monolithic model a service fault may bring down the entire monolith

* Source: "Microservices Beyond the Hype: What You Gain and What You Lose" [Merson15b]

Microservices benefits (2)

- **Modifiability**
  - The team can redesign and redeploy each microservice independently
  - The team has freedom to employ different languages, frameworks, libraries, and patterns to design and implement each microservice
  - Microservices are loosely coupled, modular components accessible only via their contracts
- **Availability**
  - Rolling out a new version of a microservice requires little downtime, whereas in the monolithic model it requires a slower restart of the entire monolith

Microservices benefits (3)

- **Scalability**
  - Each microservice can be scaled independently using pools, clusters, and grids
  - The deployment characteristics make microservices a great match for the elasticity of the cloud
- **Management**
  - Application *development* effort is divided across teams that are smaller and work more independently
- **Reusability**
  - Widespread creation of services modeled with reusability and composability in mind enable reuse beyond a single application

Microservices challenges (1)

- **Performance**
  - Services may need to communicate over the network, whereas services within the monolith may benefit from local (in-process, in-vm) calls
- **Memory use**
  - Several classes and libraries are often replicated in each microservice bundle, and the overall memory footprint increases
- **Deployability**
  - Deployment becomes more complex with many jobs, scripts, transfer areas, and configuration files

That's why fully automated deployment is required

Microservices challenges (2)

- **Modifiability**
  - Changes to the contract more likely impact consumers deployed elsewhere
    - In the monolithic model, consumers are more likely to be within the monolith and will be rolled out in lockstep with the service
  - Mechanisms to improve autonomy, such as eventual consistency and asynchronous calls, add complexity to microservices
- **Testability**
  - Automated E2E tests are harder to set up and run because they may span different microservices on different runtime environments

15

Microservices challenges (3)

- **Management**
  - The application *operation* effort increases because there are more deployed components, log files, and connections to oversee
- **Runtime autonomy**
  - In the monolith, the business logic of the system is collocated, whereas with microservices the logic is spread across microservices
  - All else being equal, a microservice is more likely to interact with other microservices over the network—that interaction decreases autonomy
- **Security**
  - More independently deployed services on the network == increased attack surface
  - Technology diversity implies multiple, diverse security mechanisms

16

17

Questions?

Paulo Merson
pmerson@cmu.edu

18