Microservices

An (Almost?) Complete Picture

Oliver Sturm • @olivers • oliver@oliversturm.com

Oliver Sturm

- Training Director at DevExpress
- Consultant, trainer, author, software architect and developer for over 25 years
- Microsoft C# MVP
- Contact: oliver@oliversturm.com

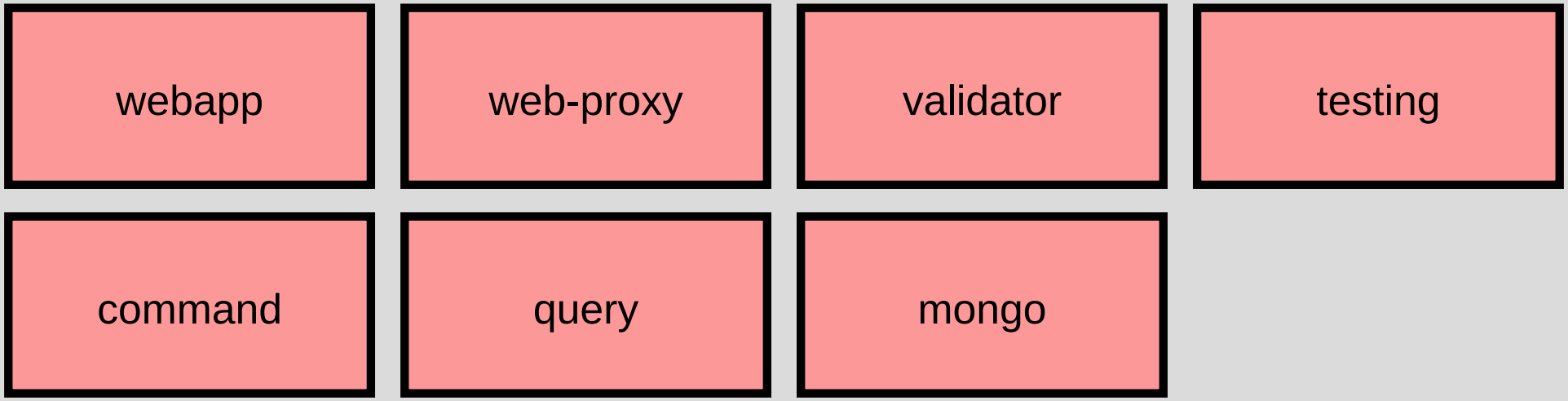
Agenda

- Service structure
  - A look at a microservices architecture
- Communication
  - Considerations pro and con frameworks
  - Working with individual services
- Packaging/deployment
  - Developer concerns
  - Real-world deployment with AWS
- Developer stuff
  - Debugging
- And what about Serverless?

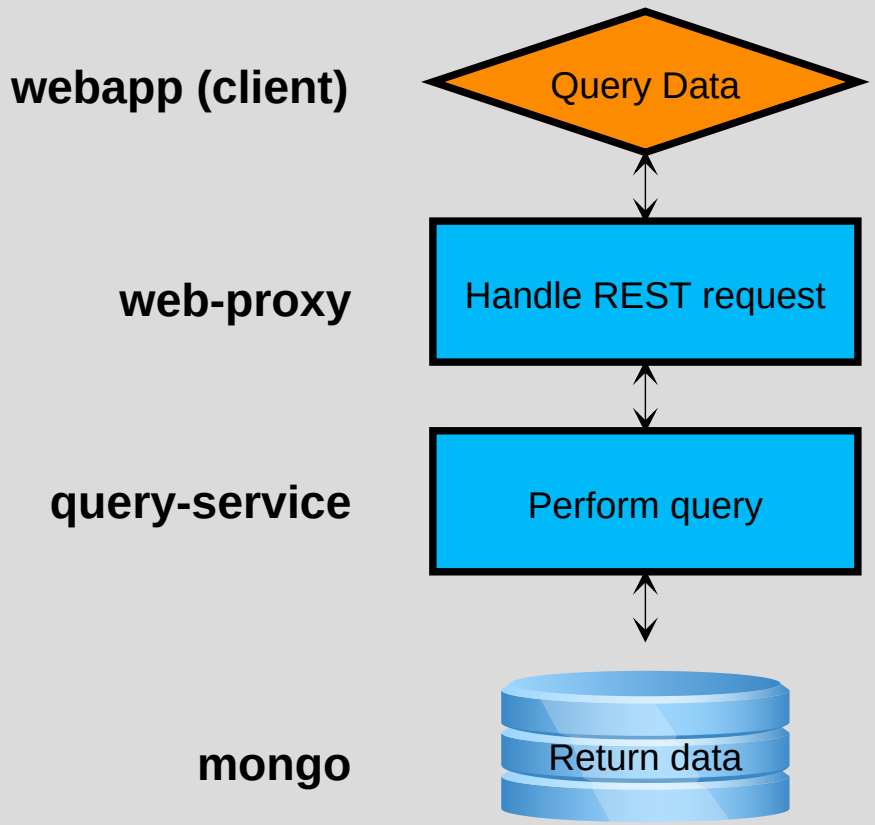
Major Goals of Microservice Architectures

- Services should be easily maintainable *building blocks*
- *Separation of concerns* and responsibilities, technically and in development teams
- Benefit from *skills* and *technological advancements*
- Long-term security for investments through *modularization*

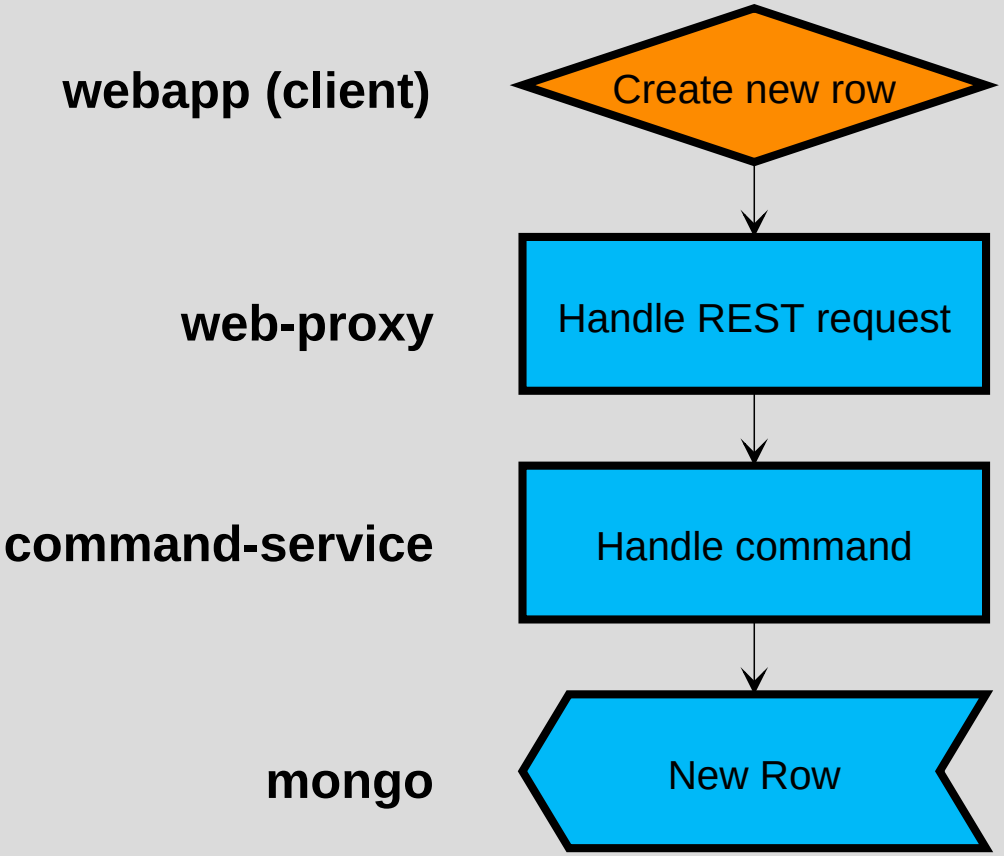
Seven Services of the Demo Application



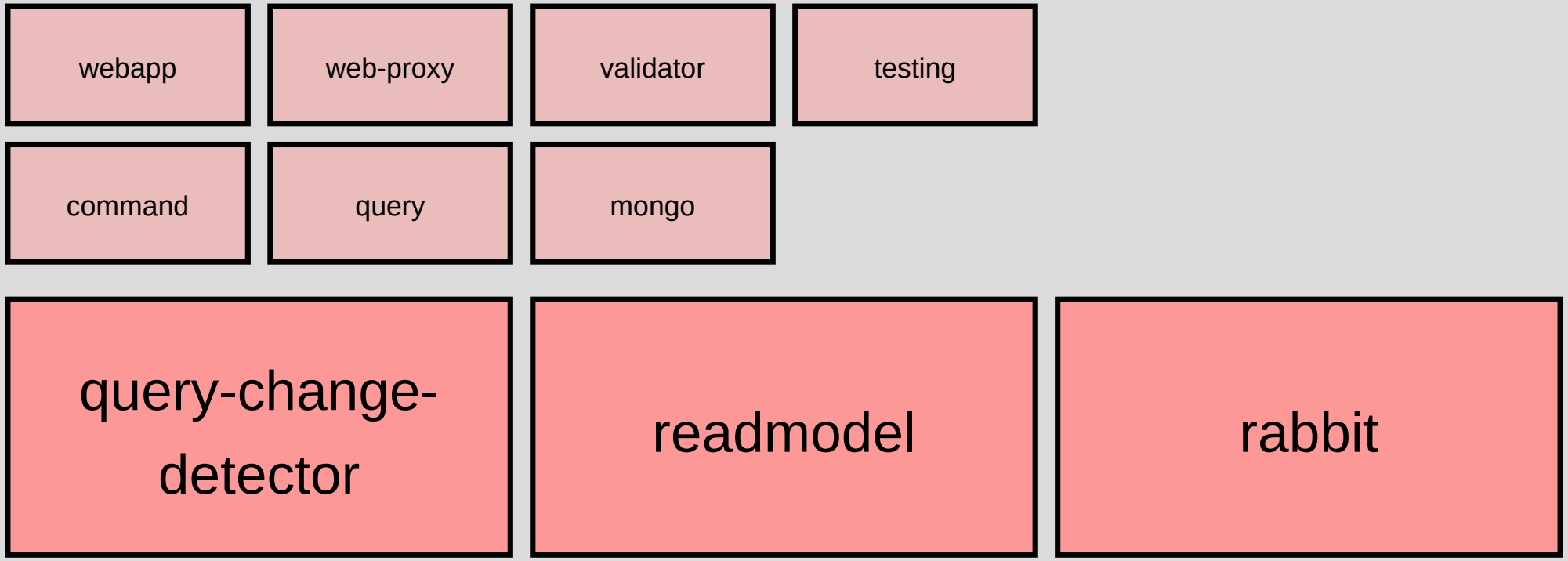
Querying Data



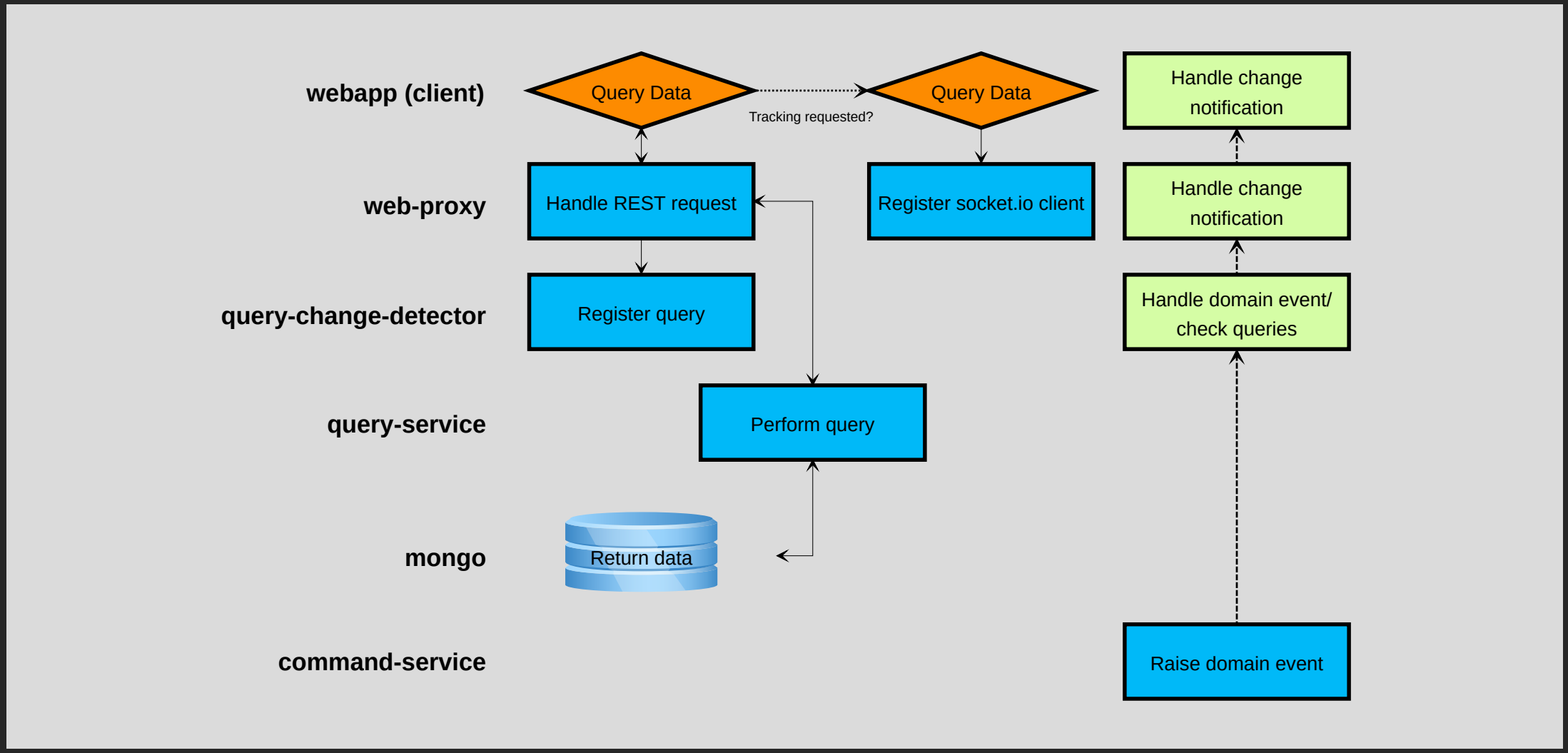
Creating a New Row



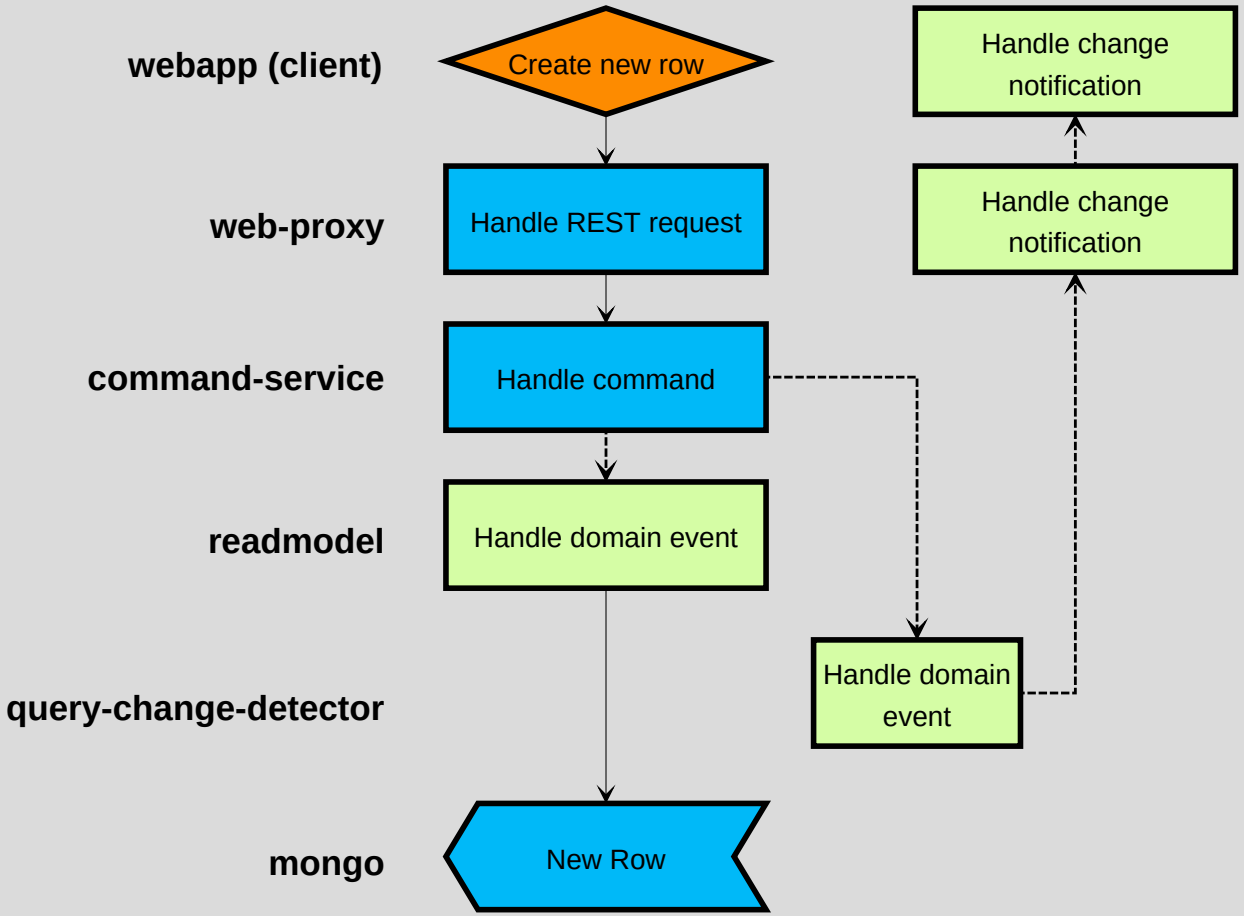
Additional Services for Advanced Architecture



Querying Data with CQRS/ES



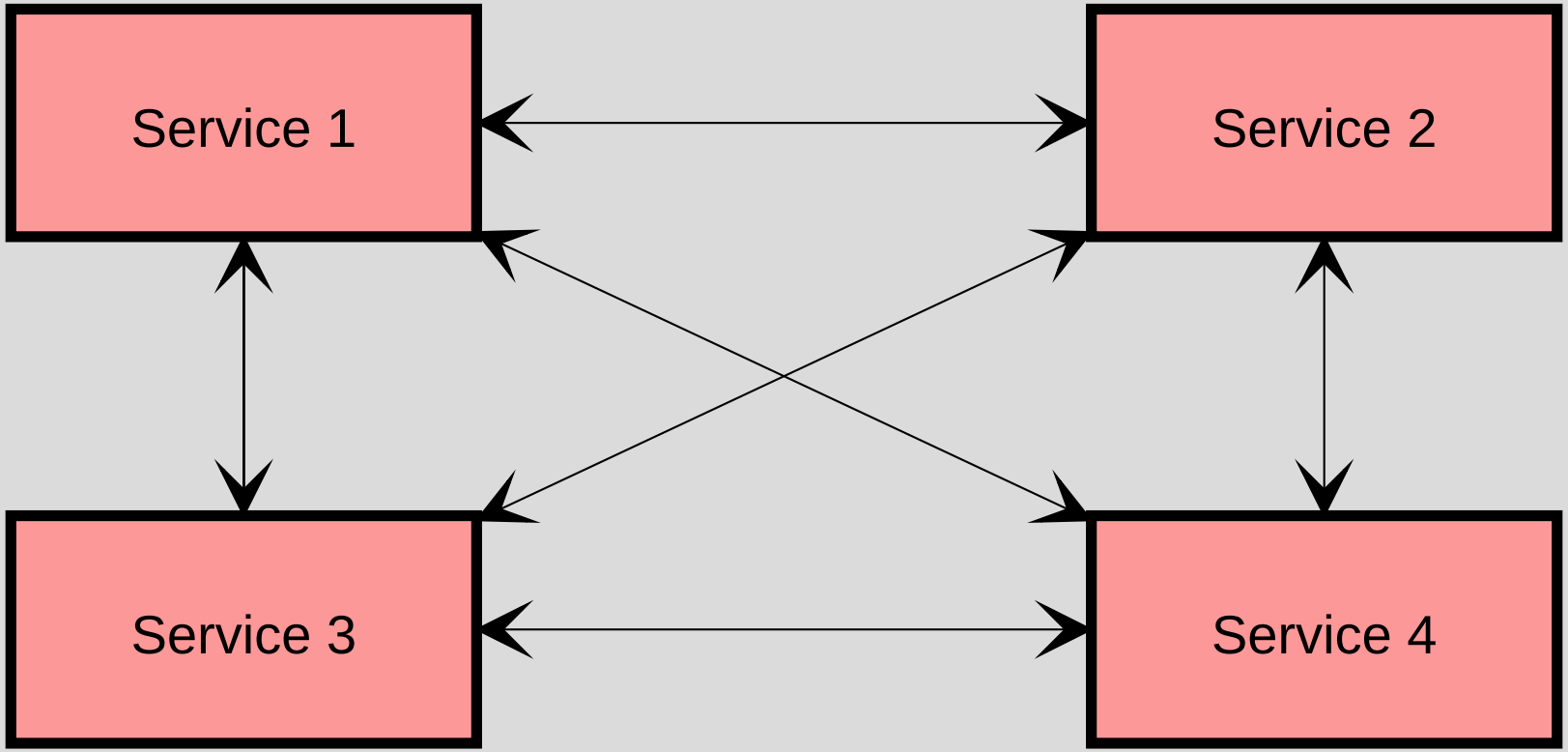
Creating a New Row with CQRS/ES



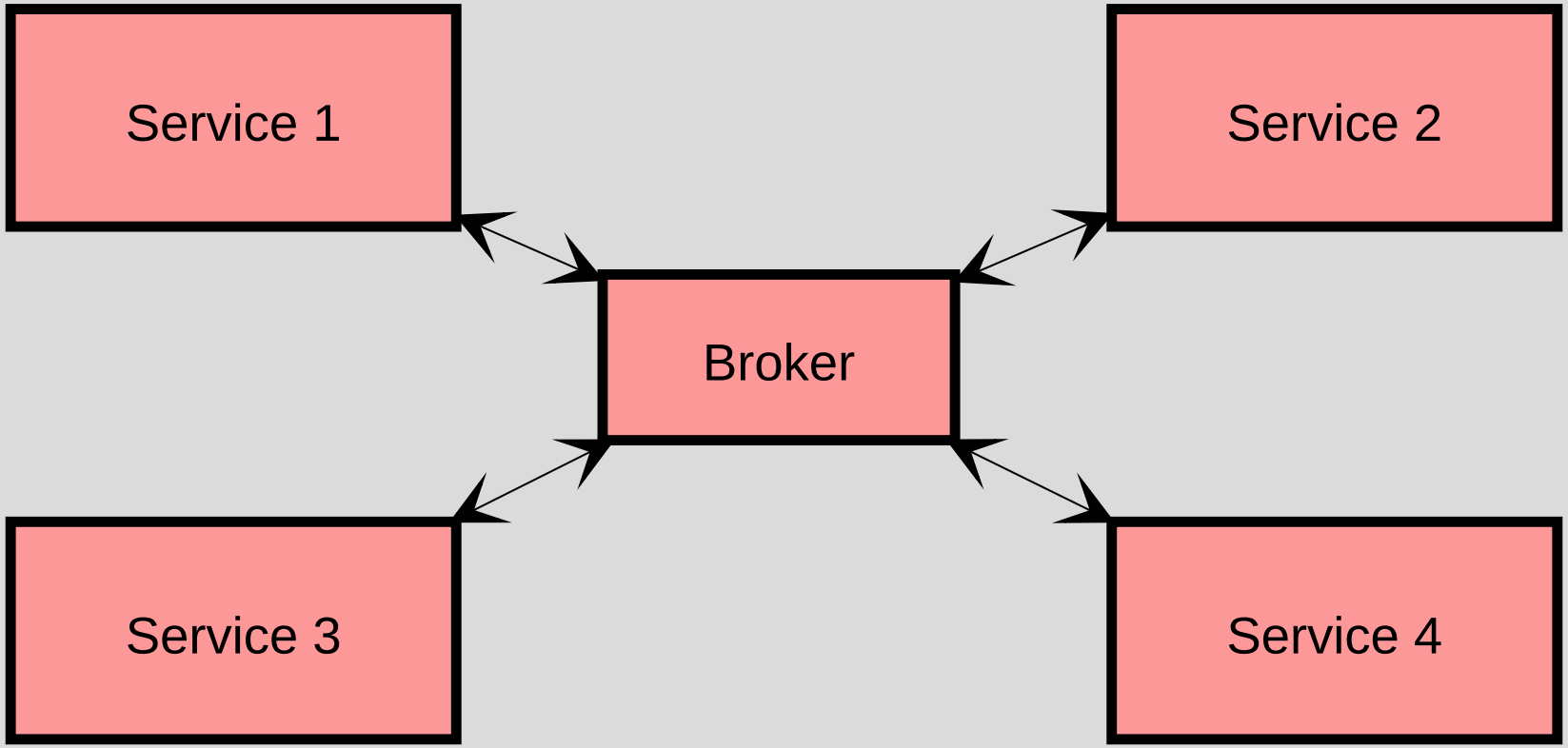
Communication

- Structural question: *who talks to who*?
- Implementation question: *how does the talking work*?

Direct Communication



Using a Broker



How does the Talking Work?

- Each service could be an *open web service* with its own external interface. REST? Proprietary? Your choice.
- Each service could be implemented to *talk to the broker exclusively*
- *Libraries* exist that implement communication

Packaging/Deployment

- Running lots of services manually isn't much fun
  - Consider *automation*
- Services may need *individual runtime environments*
- *Container systems* to the rescue!

Debugging

- *Granularity* of services makes it easy to test
- Services can be debugged as *individual autonomous entities*
  - Best regards from functional programming!

And what about Serverless?

- These considerations still apply:
  - Structural aspects
  - Communication
- Per platform, choices are made for us in order to save work
  - Communication
  - Deployment
  - General service environment
- Some aspects, like testing and debugging, get way more complicated...
- Serverless is Microservices taken to extremes — but also without some of the benefits!

Sources

- This presentation:
  - https://oliversturm.github.io/microservices-complete-picture
  - PDF download: https://oliversturm.github.io/microservices-complete-picture/slides.pdf
- Demo code:
  - https://github.com/oliversturm/cqrs-grid-demo (check *master* and *event-sourcing* branches)

Thank You

Please feel free to contact me about the content anytime.

Oliver Sturm • @olivers • oliver@oliversturm.com