HOCHSCHULE
HANNOVER
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS
–
Fakultät IV
Wirtschaft und
Informatik

CC_ITM Competence Center
Information Technology and Management
Institute at the University of Applied Sciences
and Arts Hannover | cc_itm@hs-hannover.de

Service Computation 2021
April 18 – 22, 2021 – Porto, Portugal

Towards a Microservices Reference Architecture for Insurance Companies

A. Koschel, A. Hausotter (presenter), R. Buchta, A. Grunewald, M. Lange, P. Niemann

Faculty of Business and Computer Science
University of Applied Sciences and Arts, Hannover
Ricklinger Stadtweg 120 30459 Hannover
{arne.koschel | andreas.hausotter}@hs-hannover.de

Presenter

Dr. ANDREAS HAUSOTTER is a professor emeritus for distributed information systems and database systems at the University for Applied Sciences and Arts, Hanover, Germany, Faculty of Business and Computer Science. His area of specialization comprises service computing – including service-oriented Architectures (SOA) and microservices – Java EE, webservices, distributed information systems, business process management, business rules management, and information modeling.

In 1979 he received his PhD in mathematics at Kiel University, Faculty of Mathematics and Natural Sciences. After graduation he started his career with KRUPP ATLAS ELEKTRONIK, Bremen, as a systems analyst and systems programmer in the area of real time processing. In 1984 he was hired as systems engineer and group manager SNA Communications for NIXDORF COMPUTER, Paderborn. After that, he worked for HAAS CONSULT, Hanover, as a systems engineer and product manager for traffic guidance systems.

In 1996 he was appointed professor of operating systems, networking and database systems at the University of Applied Sciences and Arts, Hanover. He has been retired since March 2018.

From the beginning he was involved in several research projects in cooperation with industry partners. During his research semester he developed a Java EE / EJB application framework. Based on this framework a web-based simulation software for securities trading was implemented by his research group to train the apprentices of the industry partner.

In 2005, the Competence Center IT & Management (CC_ITM) was founded in cooperation with industry partners. Different ambitious research projects have since then been carried out in the context of service-computing, microservices, cloud computing, business process management, and business rules management.

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

CC_ITM@HsH

CC_ITM Competence Center Information Technology and Management Institute at the University of Applied Sciences and Arts Hannover | cc_itm@hs-hannover.de

- **Competence Center Information Technology & Management (CC_ITM)**
  - Institute at the University of Applied Sciences and Arts, Hannover
  - Founded in 2005 by colleagues from the departments of **Business Information Systems and Computer Science**
  - Members: **Faculty staff**, **industry partners** (practitioners) of different areas of businesses
- Main objective
  - **Knowledge transfer** between university and industry
- Research topics
  - Management of information processing
  - Service computing, including Microservices, Service-oriented Architectures (SOA), Business Process/Rules Management (BPM/BRM)
  - Cloud Computing

Introduction

The ultimate goal of our current research is to develop a ’Microservices Reference Architecture for Insurance Companies’ jointly with our partner companies.

Questions to be answered:

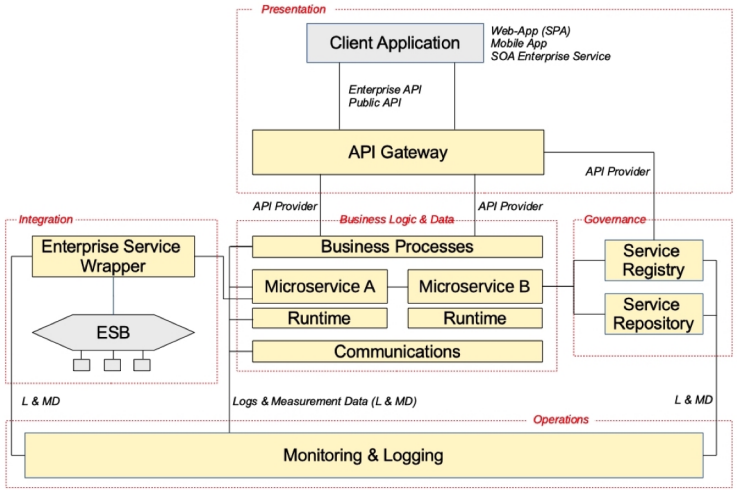
- Which information from business and technical services shall be provided for architects, developers, operators, etc.?
- How to integrate with business processes – is service orchestration or choreography (or both) more suitable for microservices?
- How to coexist with the given SOA services and their Enterprise Service Bus (ESB)?
- What about transactions and consistency?
- Compliance aspects, and more.

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

Reference Architecture for Microservices

- **Coexistence**: Legacy applications, SOA and microservices based applications will be operated in parallel for a longer transition period.
- **Observability**: A unified monitoring and logging approach has to be designed.



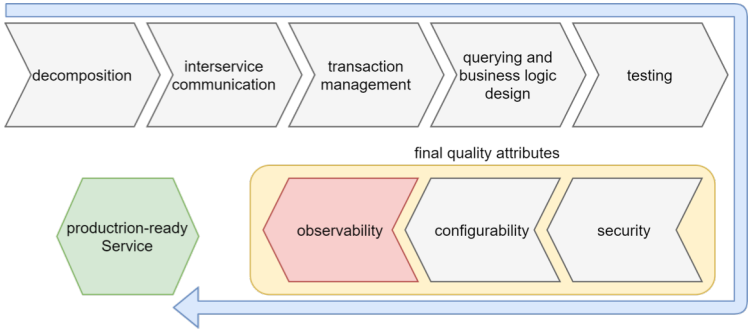
Building Blocks of the Logical Reference Architecture RaMicsV [own representation].

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

Introduction to Logging and Monitoring

- Observability is a final **quality attribute**.
- It is important to first **produce** the right data, then **collect** it and then **monitor** it.
- We are concerned with the objective of how we can create a uniform, fully comprehensive, traceable environment for monitoring and logging.



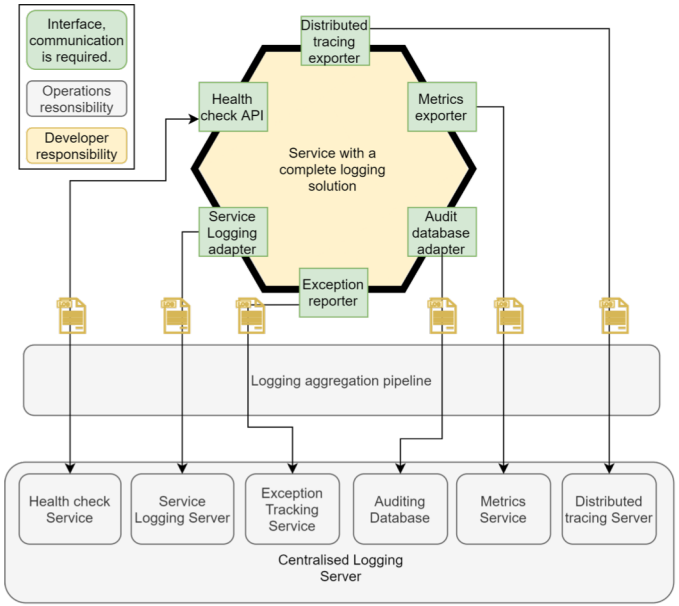
Typical building blocks for the development of a (micro)Service [own representation].

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

Patterns for Logging and Monitoring

- **Health Check API** reports on the health status of the service.
- **Log Aggregation** is important for the evaluation of distributed systems.
- **Distributed Tracing** is for the traceability of distributed requests.
- **Application Metrics** is for application insights. Here, the value generated is strongly dependent on the content.
- **Exception Tracking** separate treatment of exceptions.
- **Audit Logging** provides information about the actions taken.



Exemplary implementation of all patterns in combination [9, adopted with modifications].

Patterns for Logging and Monitoring

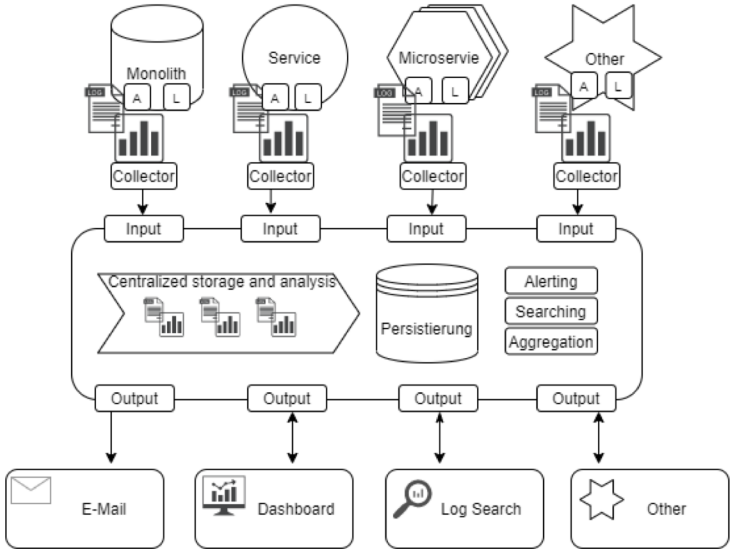
- **Not all** patterns can always be implemented in a **meaningful** way.
- The **level of detail** in which the individual patterns are implemented **varies** from application to application.
- Most of the time, a **subset of the patterns** is the right and sufficient choice to fully observe the service for the purpose it fulfills.
- If there are multiple instances of a service, log aggregation should be performed across all log types to evaluate the real behavior of the service.
- **Coordination** at the interfaces is **unavoidable**. The developers are responsible for creating decent log entries and the operators are responsible for what the users get to see in the end.
- The challenges here, are quite **similar** to that of **distributed systems** in general. Here we have a special focus on several instances.

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

Logical Reference Architecture

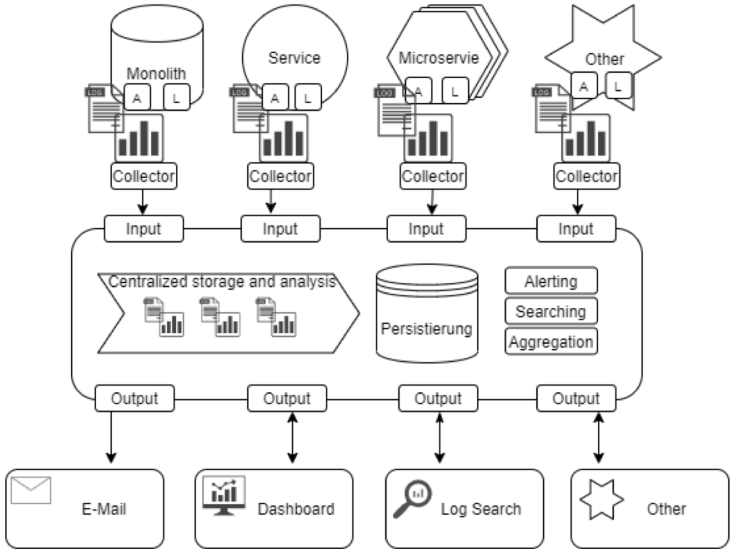
- **Agents (A):** Sort of external process to instrument processes at runtime. One of two methods are used to do so:
  - Injecting code through a external process.
  - In-process agent, that uses defined rules to trace specific actions.
- **Libraries (L):** Used in services to handle the key components for instrumentation and context propagation through a standardized API.



Logical reference architecture of the monitoring and logging environment [own representation].

Logical Reference Architecture

- **Collector:** Responsible for translating incoming data into another format, sampling and computing aggregate statistics about incoming data.
- **Centralized storage and analysis:** Gathering all telemetry data, storing and analyzing.



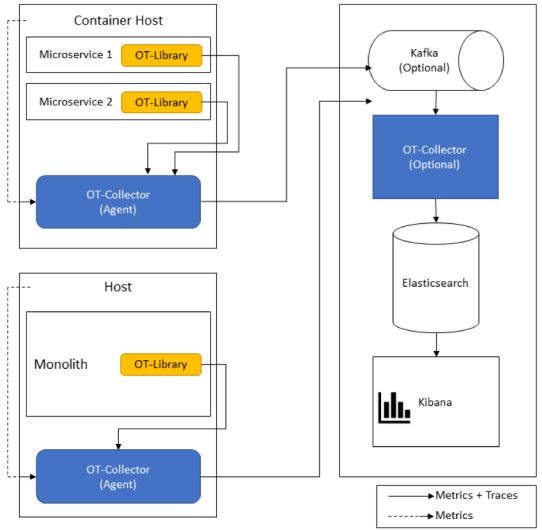
Logical reference architecture of the monitoring and logging environment [own representation].

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

Technical Reference Architecture

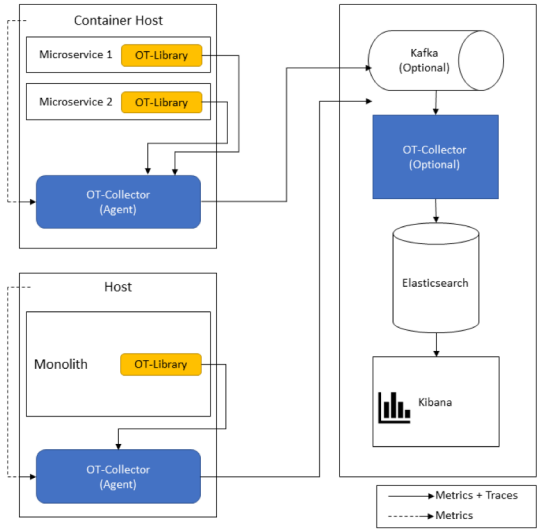
- **Open Telemetry Libraries:** Instrumentation mechanism which supports manual (code modified) and automatic (byte-code) instrumentation.
- **Open Telemetry Collector:**
  - Vendor-agnostic implementation to receive, process and export telemetry data.
  - Default location for data export
  - Can be deployed as agent or as gateway.



Technical reference architecture of the monitoring and logging environment using OpenTelemetry [own representation].

Technical Reference Architecture

- **Elasticsearch:** Distributed search and analytic engine for near real-time processing of data.
- **Kibana:** Dashboard for visualizing and analyzing data as well as managing, monitoring and securing the elastic stack.



Technical reference architecture of the monitoring and logging environment using OpenTelemetry [own representation].

Agenda

1. Introduction
2. Reference Architecture for Microservices
3. Introduction to Logging and Monitoring
4. Patterns for Logging and Monitoring
5. Logical reference architecture
6. Technical reference architecture
7. Conclusion and Future Work

Conclusion and Future Work

- In this article, we present initial steps towards a reference architecture for microservices, which we are creating jointly with our partners from the insurance industry.
- The focus was on the operations responsibility area, by presenting conceptual and technical details on logging and monitoring.
- The next steps in our research are the design the business process component and the integration responsibility area.

Thank you for your attention!

References

[1] M. Fowler and J. Lewis, “Microservices a definition of this new architectural term,” https://martinfowler.com/articles/microservices.html, March 2014, [retrieved: 3, 2021].

[2] H. Knoche and W. Hasselbring, “Drivers and barriers for microservice adoption–a survey among professionals in germany,” Enterprise Modelling and Information Systems Architectures (EMISAJ), vol. 14, 2019, p. 10.

[3] A. Hausotter, C. Kleiner, A. Koschel, D. Zhang, and H. Gehrken, “Always stay flexible! wfms-independent business process controlling in soa,” in 2011 15th IEEE Intl. Enterprise Distributed Object Computing Conference Workshops. IEEE, 2011, pp. 184–193.

[4] A. Hausotter, A. Koschel, M. Zuch, J. Busch, and J. Seewald, “Components for a SOA with ESB, BPM, and BRM – Decision framework and architectural details,” Intl. Journal On Advances in Intelligent Systems, vol. 9, no. 3,4, Dec. 2016, pp. 287– 297, [Online]. Available: https://www.thinkmind.org/index.php?view= article&articleid=intsys v9 n34 2016 6. [retrieved: 3, 2021].

[5] A. Hausotter, A. Koschel, J. Busch, and M. Zuch, “A Flexible QoS Measurement Platform for Service-based Systems,” Intl. Journal On Advances in Systems and Measurements, vol. 11, no. 3,4, Dec. 2018, pp. 269–281, [Online]. Available: https://www.thinkmind.org/index.php?view=article\&articleid= sysmea\ v11\ n34\ 2018\ 4. [retrieved: 3, 2021].

References

[6] A. Koschel, A. Hausotter, M. Lange, and P. Howeihe, “Consistency for Microservices - A Legacy Insurance Core Application Migration Example,” in SERVICE COMPUTATION 2019, The Eleventh International Conference on Advanced Service Computing, Venice, Italy, 2019, [Online]. Available: https://thinkmind.org/index.php?view= article&articleid=service computation 2019 1 10 18001. [retrieved: 3, 2021].

[7] S. Newman, Building microservices: designing fine-grained systems. Sebastopol, California: O’Reilly Media, Inc., 2015.

[8] L. Krause, Microservices: Patterns and Applications: Designing finegrained services by applying patterns. Lucas Krause, 2015.

[9] C. Richardson, Microservices Patterns: With examples in Java. Shelter Island, New York: Manning Publications, 2018.

[10] S. Angelov, P. Grefen, and D. Greefhorst, “A classification of software reference architectures: Analyzing their success and effectiveness,” in 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, IEEE, Ed., 2009.

References

[11] A. Koschel, A. Hausotter, M. Lange, and S. Gottwald, “Keep it in Sync! Consistency Approaches for Microservices - An Insurance Case Study,” in SERVICE COMPUTATION 2020, The Twelfth International Conference on Advanced Service Computing, Nice, France, 2020, [Online]. Available: http://www.thinkmind.org/index.php?view= article&articleid=service computation 2020 1 20 10016. [retrieved: 3, 2021].

[12] Y. Yu, H. Silveira, and M. Sundaram, “A microservice based reference architecture model in the context of enterprise architecture,” in 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). IEEE, 2016, pp. 1856– 1860.

[13] M. Nygard, Release It! Design and Deploy Production-Ready Software. Pragmatic Bookshelf, 2007.

[14] A. Chuvakin, K. Schmidt, and C. Phillips, Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management. Waltham, Massachusetts: Syngress Publishing, 2012.

[15] J. Turnbull, The Art of Monitoring. Turnbull Press, 2014.

[16] A. Parker, D. Spoonhower, J. Mace, B. Sigelman, and R. Isaacs, Distributed Tracing in Practice - Instrumenting, Analyzing, and Debugging Microservices. Sebastopol, California: ”O’Reilly Media, Inc.”, 2020.

References

[17] The OpenTelemetry Authors, “Documentation | OpenTelemetry,” https: //opentelemetry.io/docs/ [retrieved: 3, 2021].

[18] Elastic, “What is Elasticsearch? | Elasticsearch Reference [7.11] | Elastic,” https://www.elastic.co/guide/en/elasticsearch/reference/ current/elasticsearch-intro.html, [retrieved: 3, 2021]