Microservices Beyond COVID-19

Antonio Brogi
Department of Computer Science, University of Pisa, Italy

Abstract

This article summarises the contents of the invited keynote that I gave back in September 2020 at the "Microservices 2020" Conference, which was held entirely online during the COVID-19 pandemic.

In that keynote, I started from the question of how we can check whether a software application satisfies the main principles of microservices and –if not– of how should we refactor it. To answer that question, I discussed the capacity of existing techniques to automatically extract an architectural description of a microservice-based application, to identify architectural smells possibly violating microservices' principles, and to select suitable refactorings to resolve them. I also discussed how a (minimal) modelling of microservice-based applications can considerably simplify their design and automate their container-based deployment. Finally, I tried to point to some interesting directions for future research on microservices.

2012 ACM Subject Classification Software and its engineering

Keywords and phrases Microservice-based systems

Digital Object Identifier 10.4230/OASIcs.Microservices.2020-2022.1

Category Invited Paper

Funding Work partly funded by UNIPI PRA_2022_64 *"OSMWARE - hOlistic Sustainable Management of distributed softWARE systems"* project, funded by the University of Pisa, Italy.

Acknowledgements I would like to thank all the colleagues with whom I had the pleasure of carrying on the research activities that were described in this keynote, starting from Jacopo Soldani, with whom I shared most of this work, and continuing with (in alphabetical order) Matteo Bogo, Giuseppe Muntoni, Davide Neri, Luca Rinaldi, and Olaf Zimmermann. I would like to thank also Hernan Astudillo, Edoardo Baldini, Javier Berrocal, Giuseppe Bisicchia, Stefano Chessa, Giorgio Dell'Immagine, Stefano Forti, Marco Gaglianese, Juan Luis Herrera, Javad Khalili, Juan M. Murillo, Federica Paganelli, and Francisco Ponce, with whom I co-authored the more recent work (after Microservices 2020) cited in the last part of this article.

1 Design principles, architectural smells and refactorings

I started my keynote by recalling the main motivations and characteristics of microservices, and then I considered the following question:

> *How can architectural smells affecting design principles of microservices be detected and resolved via refactoring?*

Informally speaking, an architectural smell is a "suspect" that the defined architecture may affect a design principle. As an example of possible answer to the above question, I presented the results of the multi-vocal review [8], aimed at identifying the most recognised architectural smells for microservices, and the architectural refactorings to resolve them. That review identified seven architectural smells potentially affecting four design principles of microservices, and 13 refactoring techniques to resolve those architectural smells.

I then presented the μ`Freshener` tool [13], which automatically identifies the architectural smells present in a microservice-based application, and which allows applying architectural refactorings to resolve the identified smells.

Joint Post-proceedings of the Third and Fourth International Conference on Microservices (Microservices 2020/2022).
Editors: Gokila Dorai, Maurizio Gabbrielli, Giulio Manzonetto, Aomar Osmani, Marco Prandini, Gianluigi Zavattaro, and Olaf Zimmerman; Article No. 1; pp. 1:1–1:3
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 From incomplete specifications to running applications

I then moved to considering the question of how to select an appropriate runtime enviroment for each microservice of an application, during the design phase. As an example of possible answer to the above question, I presented the `TosKeriser` tool [3], which automatically completes a TOSCA application specifications by discovering and including Docker-based runtime environments providing the software support needed by each microservice.

I then moved to considering the question of how to suitably package each microservice into the selected runtime environment. As an example of possible answer to the above question, I presented the `TosKose` tool [2], which enables deploying microservice-based applications on top of existing container orchestrators, and to manage each service independently from the container used to run it.

3 Mining the architecture of microservice-based applications

Manually generating the description of the software architecture of an application consisting of dozens, when not hundreds, of microservices is a complex, time-consuming, and error-prone process. Software architects need to be supported by tools capable of automatically mining the software architecture of their microservice-based application.

As an example of such support, I presented the μ`Miner` tool [13], which automatically extracts the software architecture of a " black-box" microservice-based application. Without accessing the application source code, μ`Miner` derives the software architecture from the declarative specification of its Kubernetes deployment, by performing both static and dynamic analyses.

4 Concluding remarks

At the end of the keynote, I summarised the toolchain sketched in Figure 1, obtainined by pipelining the four tools described during the talk.

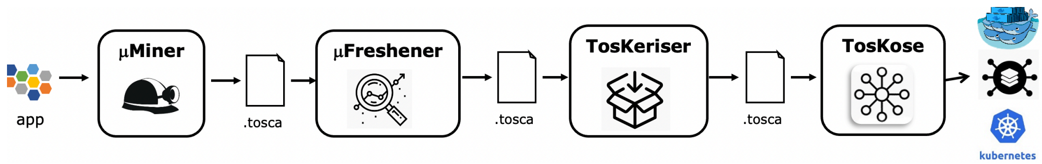

Figure 1 Toolchain example.

Take-home message: The toolchain can be taken as an example of how a (minimal) modelling of microservice-based applications can considerably simplify their design and analysis and allow automating their container-based completion and deployment.

Finally, here is a non-exhaustive list of possible interesting directions for future research on microservices on which I am working with my group and other colleagues:

- improve the techniques for *detecting and resolving architectural smells* in microservice-based applications – with alternative techniques (e.g. like [12]) for automatically extracting the software architecture of an application from its Kubernetes deployment, and for resolving architectural smells by directly modifying the Kubernetes manfiest of an application,

- improve the techniques for *detecting security smells* in microservice-based applications – by identifying the most recognised security smells for microservices (e.g. like in [9]), and by developing automated detectors (e.g. like the extensible `KubeHound` tool [4]),
- improve the tecnhiques for *determining the root causes of microservices' failures* [10] and for *explaining how failures propagate* across microservices (e.g. as in [11]),
- improve the techniques for achieving a *lightweight but effective monitoring* of microservice-based applications deployed on a distributed infrastructure [6],
- consider *sustainaibility* aspects during the entire life-cycle of microservice-based applications [1],
- develop and apply *continuous reasoning techniques* to efficently manage distributed applications in continuity with existing CI/CD pipelines and monitoring tools (e.g. like in [5, 7]).

References

1. Edoardo Baldini, Stefano Chessa, and Antonio Brogi. Estimating the environmental impact of green IoT deployments. *Sensors*, 23(3), 2023. `doi:10.3390/S23031537`.
2. Matteo Bogo, Jacopo Soldani, Davide Neri, and Antonio Brogi. Component-aware Orchestration of Cloud-based Enterprise Applications, from TOSCA to Docker and Kubernetes. *Software: Practice and Experience*, 50:1793–1821, 2020. `doi:10.1002/SPE.2848`.
3. Antonio Brogi, Davide Neri, Luca Rinaldi, and Jacopo Soldani. Orchestrating incomplete TOSCA applications with Docker. *Science of Computer Programming*, 166:194–213, 2018. `doi:10.1016/J.SCICO.2018.07.005`.
4. Giorgio Dell'Immagine, Jacopo Soldani, and Antonio Brogi. KubeHound: Detecting Microservices' Security Smells in Kubernetes Deployments. *Future Internet*, 15(7), 2023. `doi:10.3390/FI15070228`.
5. Stefano Forti, Giuseppe Bisicchia, and Antonio Brogi. Declarative Continuous Reasoning in the Cloud-IoT Continuum. *Journal of Logic and Computation*, 32(2):206–232, 2022. `doi:10.1093/LOGCOM/EXAB083`.
6. Marco Gaglianese, Stefano Forti, Federica Paganelli, and Antonio Brogi. Assessing and enhancing a Cloud-IoT monitoring service over federated testbeds. *Future Generation Computer Systems*, 147:77–92, 2023. `doi:10.1016/J.FUTURE.2023.04.026`.
7. Juan Luis Herrera, Javier Berrocal, Stefano Forti, Antonio Brogi, and Juan M. Murillo. Continuous QoS-Aware Adaptation of Cloud-IoT Application Placements. *Computing*, 105:2037–2059, 2023. `doi:10.1007/S00607-023-01153-1`.
8. Davide Neri, Jacopo Soldani, Olaf Zimmermann, and Antonio Brogi. Design principles, architectural smells and refactorings for microservices: A multivocal review. *Software-Intensive Cyber-Physical Systems*, 35:3–15, 2020. `doi:10.1007/S00450-019-00407-8`.
9. Francisco Ponce, Jacopo Soldani, Hernan Astudillo, and Antonio Brogi. Smells and Refactorings for Microservices Security: A Multivocal Literature Review. *Journal of Systems & Software*, 4(C), 2023. `doi:10.1016/J.JSS.2022.111393`.
10. Jacopo Soldani and Antonio Brogi. Anomaly Detection and Failure Root Cause Analysis in (Micro)Service-Based Cloud Applications. *ACM Computing Surveys*, 55(3):1–39, 2022. `doi:10.1145/3501297`.
11. Jacopo Soldani, Stefano Forti, and Antonio Brogi. yRCA: An explainable failure root cause analyser. *Science of Computer Programming*, 230, 2023. `doi:10.1016/J.SCICO.2023.102997`.
12. Jacopo Soldani, Javad Khalili, and Antonio Brogi. Offline Mining of Microservice-Based Architectures. *SN Computer Science*, 4, 2023. `doi:10.1007/S42979-023-01721-4`.
13. Jacopo Soldani, Giuseppe Muntoni, Davide Neri, and Antonio Brogi. The μTOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures. *Software: Practice and Experience*, 51(7):1591–1621, 2021. `doi:10.1002/SPE.2974`.