Microservices

Microservices is a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services should be fine-grained and the protocols should be lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity and makes the application easier to understand, develop and test. It also parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently.[1] It also allows the architecture of an individual service to emerge through continuous refactoring.[2] Microservices-based architectures enable continuous delivery and deployment.[3]

Contents

Details

There is no industry consensus yet regarding the properties of microservices, and an official definition is missing as well. Some of the defining characteristics that are frequently cited include:

- Services in a microservice architecture (MSA)[4] are often processes that communicate with each other over a network in order to fulfill a goal using technology-agnostic protocols such as HTTP.[5][6][7][8] However, services might also use other kinds of inter-process communication mechanisms such as shared memory.[9] Services might also run within the same process as, for example, OSGI bundles.
- Services in a microservice architecture should be independently deployable.[10]
- The services are easy to replace.
- Services are organized around capabilities, e.g., user interface front-end, recommendation, logistics, billing, etc.
- Services can be implemented using different programming languages, databases, hardware and software environment, depending on what fits best.
- Services are small in size, messaging enabled, bounded by contexts, autonomously developed, independently deployable, decentralized and built and released with automated processes.[10]

A microservices-based architecture:

- Naturally enforces a modular structure.[6]

- Lends itself to a continuous delivery software development process. A change to a small part of the application only requires one or a small number of services to be rebuilt and redeployed.[6]
- Adheres to principles such as fine-grained interfaces (to independently deployable services), business-driven development (e.g. domain-driven design), IDEAL cloud application architectures, polyglot programming and persistence, lightweight container deployment, decentralized continuous delivery, and DevOps with holistic service monitoring.[11]
- Provides characteristics that are beneficial to scalability.[12][13][14]

History

A workshop of software architects held near Venice in May 2011 used the term "microservice" to describe what the participants saw as a common architectural style that many of them had been recently exploring. In May 2012, the same group decided on "microservices" as the most appropriate name. James Lewis presented some of those ideas as a case study in March 2012 at 33rd Degree in Kraków in Microservices - Java, the Unix Way, as did Fred George about the same time. Adrian Cockcroft at Netflix, describing this approach as "fine grained SOA", pioneered the style at web scale, as did many of the others mentioned in this article - Joe Walnes, Dan North, Evan Bottcher and Graham Tackley.[15]

Dr. Peter Rodgers introduced the term "Micro-Web-Services" during a presentation at the Web Services Edge conference in 2005. On slide #4 of the conference presentation, he states that "Software components are Micro-Web-Services".[16] Juval Löwy had similar precursor ideas about classes being granular services, as the next evolution of Microsoft architecture.[17][18][19] "Services are composed using Unix-like pipelines (the Web meets Unix = true loose-coupling). Services can call services (+multiple language run-times). Complex service-assemblies are abstracted behind simple URI interfaces. Any service, at any granularity, can be exposed." He described how a well-designed service platform "applies the underlying architectural principles of the Web and Web services together with Unix-like scheduling and pipelines to provide radical flexibility and improved simplicity by providing a platform to apply service-oriented architecture throughout your application environment".[20] The design, which originated in a research project at Hewlett Packard Labs, aims to make code less brittle and to make large-scale, complex software systems robust to change.[21] To make "Micro-Web-Services" work, one has to question and analyze the foundations of architectural styles (such as SOA) and the role of messaging between software components in order to arrive at a new general computing abstraction.[22] In this case, one can think of resource-oriented computing (ROC) as a generalized form of the Web abstraction. If in the Unix abstraction "everything is a file", in ROC, everything is a "Micro-Web-Service". It can contain information, code or the results of computations so that a service can be either a consumer or producer in a symmetrical and evolving architecture.

Microservices is a specialization of an implementation approach for service-oriented architectures (SOA) used to build flexible, independently deployable software systems.[23] The microservices approach is a first realisation of SOA that followed the introduction of DevOps and is becoming more popular for building continuously deployed systems.[24]

Philosophy

The philosophy of the microservices architecture essentially equals to the Unix philosophy of "Do one thing and do it well". It is described as follows:[25][26][27]

- The services are small - fine-grained to perform a single function.
- The organization culture should embrace automation of testing and deployment. This eases the burden on management and operations and allows for different development teams to work on independently deployable units of code.[28]
- The culture and design principles should embrace failure and faults, similar to anti-fragile systems.
- Each service is elastic, resilient, composable, minimal, and complete.[27]

Linguistic approach

A linguistic approach to the development of microservices[29] focuses on selecting a programming language which can easily represent a microservice as a single software artifact. A language well suited to microservices will encourage the developer to think and program in terms of microservices from the start instead of starting with a different programming paradigm only to refactor or retrofit a finished project so it can be deployed as a microservice.[30] When effective, the gap between architecting a project and deploying it can be minimized.

One language intended to fill this role is Jolie.[31][32]

Criticism

The microservices approach is subject to criticism for a number of issues:

- Services form information barriers[33]
- Inter-service calls over a network have a higher cost in terms of network latency and message processing time than in-process calls within a monolithic service process[6]
- Testing and deployment are more complicated[34]
- Moving responsibilities between services is more difficult. It may involve communication between different teams, rewriting the functionality in another language or fitting it into a different infrastructure[6]
- Viewing the size of services as the primary structuring mechanism can lead to too many services when the alternative of internal modularization may lead to a simpler design.[35]

Cognitive load

The architecture introduces additional complexity and new problems to deal with, such as network latency, message formats, load balancing and fault tolerance.[36][34]

The complexity of a monolithic application is only shifted into the network, but persists:

> You can move it about but it's still there!
>
> — Robert Annett: Where is the complexity?[37]

Also, an application made up of any number of microservices has to access its respective ecosystem which may have unnecessary complexity.[38] This kind of complexity can be reduced by standardizing the access mechanism. The Web as a system standardized the access mechanism by retaining the same access mechanism between browser and application resource over the last 20 years. Using the number of Web pages indexed by Google it grew from 26 million pages in 1998 to around 60 trillion individual pages by 2015 without the need to change its access mechanism. The Web itself is an example that the complexity inherent in traditional monolithic software systems can be overcome.[39][40]

Nanoservices

Too-fine-grained microservices have been criticized as an anti-pattern, dubbed a **nanoservice** by Arnon Rotem-Gal-Oz:

> [A] nanoservice is an anti-pattern where a service is too fine grained. [A] nanoservice is a service whose overhead (communications, maintenance etc.) outweighs its utility.[41][42]

Problems include the code overhead (interface definition, retries), runtime overhead (serialization/deserialization, network traffic), and fragmented logic (useful functionality not implemented in one place, instead requiring combining many services).

Proposed alternatives to nanoservices include:[42]

- Package the functionality as a software library, rather than a service.
- Combine the functionality with other functionalities, producing a more substantial, useful service.
- Refactor the system, putting the functionality in other services or redesigning the system.

See also

- Conway's law
- Cross-cutting concern
- Fallacies of distributed computing
- Representational state transfer (REST)
- Service-oriented architecture (SOA)
- Unix philosophy
- Self-contained Systems
- Serverless computing
- Web-oriented architecture (WOA)

References

1. Richardson, Chris. "Microservice architecture pattern" (http://microservices.io/patterns/microservices.html). *microservices.io*. Retrieved 2017-03-19.
2. Chen, Lianping; Ali Babar, Muhammad (2014). *Towards an Evidence-Based Understanding of Emergence of Architecture through Continuous Refactoring in Agile Software Development* (https://dx.doi.org/10.1109/WICSA.2014.45). The 11th Working IEEE/IFIP Conference on Software Architecture(WICSA 2014) (https://web.archive.org/web/20140730053f454/http://wicsa2014.org/). IEEE.
3. ceracm (2017-01-04). "Microservices-based architectures enable continuous delivery/deployment" (https://blog.eventuate.io/2017/01/04/the-microservice-architecture-is-a-means-to-an-end-enabling-continuous-deliverydeployment/). *Eventuate.IO*. Retrieved 2017-03-19.
4. Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, Larisa Safina. "Microservices: yesterday, today, and tomorrow" (https://arxiv.org/pdf/1606.04036.pdf) (PDF). Retrieved 2 May 2017.
5. "Microservices" (https://martinfowler.com/articles/microservices.html). *martinfowler.com*. Retrieved 2017-02-06.
6. Martin Fowler. "Microservices" (http://martinfowler.com/articles/microservices.html).
7. Newman, Sam. *Building Microservices*. O'Reilly Media. ISBN 978-1491950357.
8. Wolff, Eberhard. *Microservices: Flexible Software Architectures* (http://microservices-book.com). ISBN 978-0134602417.
9. "Micro-services for performance" (https://vanilla-java.github.io/2016/03/22/Micro-services-for-performance.html). *Vanilla Java*. Retrieved 2017-03-19.
10. Nadareishvili, I., Mitra, R., McLarty, M., Amundsen, M., Microservice Architecture: Aligning Principles, Practices, and Culture, O'Reilly 2016
11. "IFS: Microservices Resources and Positions" (https://www.ifs.hsr.ch/index.php?id=15266&L=4). *hsr.ch*. Retrieved 28 December 2016.
12. Nicola Dragoni, Schahram Dustdar, Stephan T. Larsen, Manuel Mazzara. "Microservices: Migration of a Mission Critical System" (https://arxiv.org/pdf/1704.04173.pdf) (PDF). Retrieved 2 May 2017.

13. Nicola Dragoni, Ivan Lanese, Stephan Thordal Larsen, Manuel Mazzara, Ruslan Mustafin, Larisa Safina. "Microservices: How To Make Your Application Scale" (https://arxiv.org/pdf/1702.07149.pdf) (PDF). Retrieved 2 May 2017.
14. Manuel Mazzara, Kevin Khanda, Ruslan Mustafin, Victor Rivera, Larisa Safina, Alberto Sillitti. "Microservices Science and Engineering" (https://arxiv.org/pdf/1706.07350.pdf) (PDF). Retrieved 14 November 2017.
15. James Lewis and Martin Fowler. "Microservices" (http://martinfowler.com/articles/microservices.html).
16. Rodgers, Peter. "Service-Oriented Development on NetKernel- Patterns, Processes & Products to Reduce System Complexity Web Services Edge 2005 East: CS-3" (http://www.cloudcomputingexpo.com/node/80883). *CloudComputingExpo 2005*. SYS-CON TV. Retrieved 3 July 2017.
17. Löwy, Juval (October 2007). "Every Class a WCF Service" (https://channel9.msdn.com/Shows/ARCast.TV/ARCastTV-Every-Class-a-WCF-Service-with-Juval-Lowy). *Channel9, ARCast.TV*.
18. Löwy, Juval (2007). *Programming WCF Services 1st Edition*. pp. 543–553.
19. Löwy, Juval (May 2009). "Every Class As a Service" (https://blogs.msdn.microsoft.com/drnick/2009/04/29/wcf-at-teched-2009/). *Microsoft TechEd Conference, SOA206*. Archived from the original (https://www.youtube.com/watch?v=w-Hxc6uWCPg) on 2010.
20. Rodgers, Peter. "Service-Oriented Development on NetKernel- Patterns, Processes & Products to Reduce System Complexity" (http://www.cloudcomputingexpo.com/node/80883). *CloudComputingExpo*. SYS-CON Media. Retrieved 19 August 2015.
21. Russell, Perry; Rodgers, Peter; Sellman, Royston (2004). "Architecture and Design of an XML Application Platform" (http://www.hpl.hp.com/techreports/2004/HPL-2004-23.html). *HP Technical Reports*. p. 62. Retrieved 20 August 2015.
22. Hitchens, Ron (Dec 2014). Swaine, Michael, ed. "Your Object Model Sucks". *PragPub Magazine*. Pragmatic Programmers: 15.
23. Pautasso, Cesare (2017). "Microservices in Practice, Part 1: Reality Check and Service Design" (http://ieeexplore.ieee.org/document/7819415/). *IEEE Software*. **34** (1): 91–98. doi:10.1109/MS.2017.24 (https://doi.org/10.1109%2FMS.2017.24).
24. "Continuous Deployment: Strategies" (https://www.javacodegeeks.com/2014/12/continuous-deployment-strategies.html). *javacodegeeks.com*. Retrieved 28 December 2016.
25. Lucas Krause. *Microservices: Patterns and Applications*. ASIN B00VJ3NP4A (https://www.amazon.com/dp/B00VJ3NP4A).
26. Lucas Krause. "Philosophy of Microservices?" (http://microservicesbook.io/the-philosophy-of-microservice-architecture/).
27. Jim Bugwadia. "Microservices: Five Architectural Constraints" (http://nirmata.com/2015/02/microservices-five-architectural-constraints/).
28. Li, Richard. "Microservices Essentials for Executives: The Key to High Velocity Software Development" (https://www.datawire.io/microservices-essentials-executives-key-high-velocity-software-development/). *Datawire*. Datawire, Inc. Retrieved 21 October 2016.
29. Claudio Guidi. "What is a microservice? (from a linguistic point of view)" (http://claudioguidi.blogspot.it/2017/03/what-microservice-from-linguisitc.html).
30. Claudio Guidi, Ivan Lanese, Manuel Mazzara, Fabrizio Montesi. "Microservices: a Language-based Approach" (https://arxiv.org/abs/1704.08073). Retrieved 2 May 2017.
31. Jolie Team. "Vision of microservices revolution" (http://www.jolie-lang.org/vision.html).
32. Fabrizio Montesi. "Programming Microservices with Jolie - Part 1: Data formats, Proxies, and Workflows" (http://fmontesi.github.io/2015/02/06/programming-microservices-with-jolie.html).
33. Jan Stenberg (11 August 2014). "Experiences from Failing with Microservices" (http://www.infoq.com/news/2014/08/failing-microservices).
34. "Developing Microservices for PaaS with Spring and Cloud Foundry" (http://www.infoq.com/presentations/microservices-pass-spring-cloud-foundry).
35. Tilkov, Stefan (17 November 2014). "How small should your microservice be?" (https://www.innoq.com/blog/st/2014/11/how-small-should-your-microservice-be/). *innoq.com*. Retrieved 4 January 2017.

36. Pautasso, Cesare (2017). "Microservices in Practice, Part 2: Service Integration and Sustainability" (http://ieeexplore.ieee.org/document/7888407/). *IEEE Software*. **34** (2): 97–104. doi:10.1109/MS.2017.56 (https://doi.org/10.1109%2FMS.2017.56).
37. Robert Annett. "Where is the complexity?" (http://www.codingthearchitecture.com/2014/05/01/where_is_the_complexity.html).
38. "BRASS Building Resource Adaptive Software Systems". U.S. Government. DARPA. April 7, 2015. "Access to system components and the interfaces between clients and their applications, however, are mediated via a number of often unrelated mechanisms, including informally documented application programming interfaces (APIs), idiosyncratic foreign function interfaces, complex ill-understood model definitions, or ad hoc data formats. These mechanisms usually provide only partial and incomplete understanding of the semantics of the components themselves. In the presence of such complexity, it is not surprising that applications typically bake-in many assumptions about the expected behavior of the ecosystem they interact with."
39. Alpert, Jesse; Hajaj, Nissan. "We knew the web was big" (http://googleblog.blogspot.co.at/2008/07/we-knew-web-was-big.html). *Official Google Blog*. Google.com. Retrieved 22 August 2015.
40. "The Story" (http://www.google.com/insidesearch/howsearchworks/thestory/). *How search works*. Google.com. Retrieved 22 August 2015.
41. Services, Microservices, Nanoservices – oh my! (http://arnon.me/2014/03/services-microservices-nanoservices/), Arnon Rotem-Gal-Oz
42. Pratical SOA: 1.1: Nanoservices (http://arnon.me/wp-content/uploads/2010/10/Nanoservices.pdf), Arnon Rotem-Gal-Oz, 2010

Further reading

- Microservices – Definition, Principles and Benefits (http://howtodoinjava.com/microservices/microservices-definition-principles-benefits/)
- S. Newman, Building Microservices – Designing Fine-Grained Systems, O'Reilly, 2015
- E. Wolff, Microservices: Flexible Software Architecture (http://microservices-book.com/), Addison-Wesley, 2016
- I. Nadareishvili et al., Microservices Architecture – Aligning Principles, Practices and Culture, O'Reilly, 2016, http://transform.ca.com/rs/117-QWV-692/images/CA%20Technologies%20-%20OReilly%20Microservice%20Architecture%20eBook.pdf
- C: Richardson, Microservice architecture pattern language http://microservices.io/
- SEI SATURN 2015 microservices workshop, https://github.com/michaelkeeling/SATURN2015-Microservices-Workshop
- Wijesuriya, Viraj Brian (2016-08-29) *Microservice Architecture, Lecture Notes, University of Colombo School of Computing, Sri Lanka (http://www.slideshare.net/tyrantbrian/microservice-architecture-65505794)*
