Case Study on Building Microservices from Day 1

[Project Final Report]

Francisco Servant
fservant@vt.edu

Anand J Bangad
ajbangad@vt.edu

Kapil Kale
kapilk@vt.edu

ABSTRACT

Microservices is a buzzword in the software development sector and there is a general hype surrounding the trend in the industry to move towards this architectural style. Microservices offer a great value and added advantages. This is also evident from the case studies of companies like Netflix, Walmart and Amazon who have adopted this architecture. We know microservices are fruitful so shouldn't we begin it from day 1? We address this question through this research paper. In this paper we present opinions and perspectives of developers who have begun using microservices from day one. We also present opinions of developers who prefer using monoliths in the start stage, and as time progresses could move onto microservices or stay with their original architecture. Through their opinions and responses we present various use cases whether microservices could be used from day 1 or not. We also present an analysis on the results of survey where we asked MS students their experiences with microservices and if they have would have incorporated it from the very start.

KEYWORDS

Microservices, Monoliths, Microservices Day 1, Advantages and Disadvantages of Microservices, Software Architecture

1 INTRODUCTION

Microservices is an architectural style, an approach to software development in which a large application is built as a suite of modular services; small, independently versioned, and scalable customer-focused services with specific business goals, which communicate with each other over standard protocols with well-defined interfaces [1]. As they are independently deployable and scalable, each service also provides a firm module boundary, even allowing for different services to be written in different programming languages and can also be managed by different teams.

In software engineering, a monolithic application describes a single-tiered software application in which the user interface and data access code are combined into a single program from a single platform. A monolithic application is self-contained, and independent from other computing applications. The design philosophy is that the application is responsible not just for a particular task, but can perform every step needed to complete a particular function.

Microservices offer both advantages and disadvantages. Through developer's points of views and our research, we have enlisted these advantages and disadvantages. We also present developers opinions on whether microservices should be started from day one.

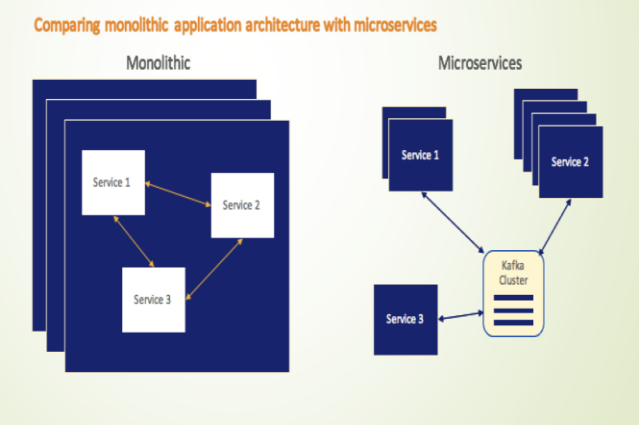

Figure1: Comparing Monoliths and Microservices

2 PROBLEM DEFINITION

The question to use microservices from day one is very debatable. The architecture and use cases are application-centric. Here, through our gathered research we provide developers information and knowledge whether to embrace or not microservices from day 1. This research will help them in making an informed decision.

3 RELATED WORK

There hasn't been much work done on this topic. Microservices is comparatively a new concept. There is no right answer or conclusion found to this question. It still relies on people's choices, preferences, experience, and expertise.

There are few papers which submit the report after doing the migration on Cloud-Native Architecture. Migrating monolithic architectures to cloud-native architectures like microservices brings in many benefits such as the flexibility to adapt to the technological changes and independent resource management for

different system components [16]. But this work has neither mentioned any issues faced during migration nor it has described disadvantages after doing the migration. Research done by Dr. Tobias Schneider [17] mentions why the microservices based architecture was chosen to do the deployment of software for connected car solutions. It also highlights the key points in using microservices such as Scalability, Exchangeability, Reuse and Continuous Deployment. The work done in the paper [18] on Microservice Architecture Enables DevOps. In this paper, the author has used asynchronous communication but has not mentioned the reason for doing so. The work done by many companies like NetFlix, Wal-Mart has got us interested in this article.

4 BACKGROUND

Here we present the actual differences between microservices and monoliths. We also demonstrate a visual difference between the two and give a side by side comparison. We present a use case of booking a hotel room and show what a microservice architecture and monolithic architecture will look like.

Let us consider there is an online website for booking a hotel. Some services that they need are Payment, Room Availability, Photos of rooms, Review by users, Fare calculation etc. Here in a microservice architecture we break down all these services into individual independent ones. Services like payment, Availability check, storing customer details are all separate services. They communicate with each other whenever needed. For communication we use either rest endpoints or message queuing systems like RabbitMQ or Kafka which are becoming very popular. In Figure 2 all the services are arranged as stacks. These stacks represent how many resources are allocated to a particular service. Let us say majority customers are checking availability on our site, then we increase the resources spent on only availability. If this were a monolithic application we would have to increase resources for all the services. This property of microservices proves to be more cost effective.

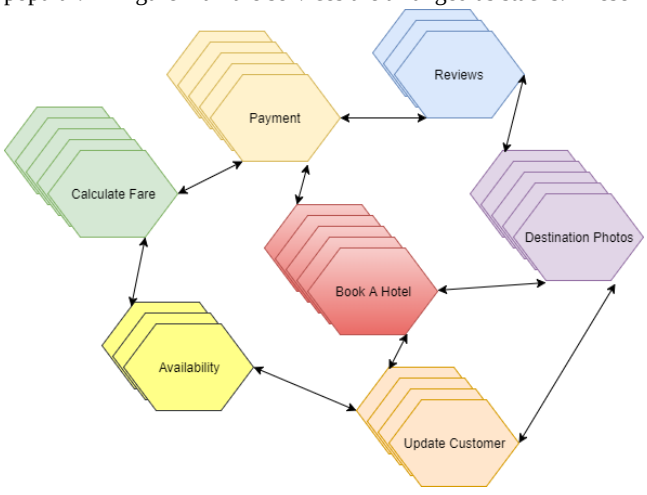

Figure2: Microservices Architecture for booking a Hotel Room

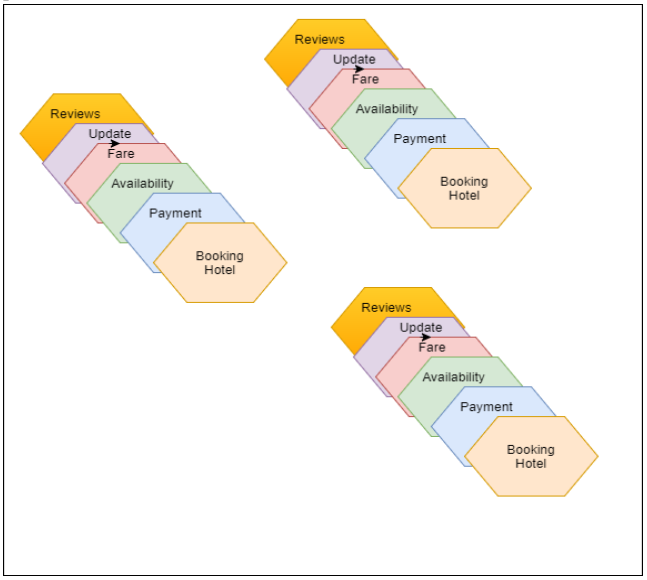

Figure3: Monolithic Architecture for booking a Hotel Room

In Figure 3 we show the architecture of a monolithic application. All the services hold dependencies with each other. If one of the service malfunctions or is down then my entire application goes down. This is not the case with microservices. If my availability service is down users can still view photos, write reviews and make payment.

5 INDUSTRY ADOPTION: MICROSERVICES

More and more companies are starting to migrate their monolith application infrastructure to a set of smaller, more manageable services, or are continuing to do so, some prominent earlier adopters stand out.

Amazon was the earliest adopter of Microservices. Amazon's CTO Werner Vogels explains their evolution from a monolithic web application called Obidos to a large-scale service-oriented architecture. Obidos, held "all the business logic, all the display logic, and all the functionality for Amazon's web shop, had undergone years of extensions and performance tuning until Amazon came to the conclusion in 2001 that its monolith based architecture would simply not scale anymore." The same held true

not just for their actual web application, but also for their efforts to optimize and scale their backing database resources.

Amazon's process for developing and launching new features relies heavily on the use of small, isolated services. Amazon has built an infrastructure in which services are so commoditized that they can be easily spun up, soft-launched, measured, and iterated upon, until the problem domain and the new features business impact are well understood.

Netflix is another company who has made early adoptions into microservices. Much of the tooling they developed in-house has been open sourced under the moniker of Netflix OSS. The Netflix approach to service versioning is to run several instances of the same service simultaneously, but with distinct versions. Instances of older versions are shut down only when all the clients have been convinced that they need to migrate off from an older version. This approach would have been very expensive if running on a monolith application.

The engineering team at SoundCloud (https://soundcloud.com/) made a similar move from a monolithic Ruby on Rails application to microservices written in Clojure, Scala, and Ruby.

Uber in 2016, took a decision to abandon its monolithic codebase in favor of a modular, flexible microservice architecture [4]. Since then, they have devoted many thousands of engineering hours to expanding this ecosystem of Uber microservices written in a variety of languages and using many different frameworks.

6 WHY USE MICROSERVICES?

After studying case studies of Uber, Amazon, Netflix, and Wal-Mart, we have compiled the advantages that microservices offers.

- Independent Deployment - Releasing software is a quarterly event or a monthly event for many companies. By using microservices many teams have cut down on their release cycles so that releases could now happen several times per day. The main advantage that this offers is increased speed of responding to market needs and staying ahead of competitors. In case of Monoliths, this is not possible as a small fraction of code change can affect the entire application. This advantage has been observed with all the above companies.
- Technology Diversity - With microservices different services can use different languages and technology to be built. Python is great for data science, NodeJS is good for server and client side communications. The power of all technologies can be used and the teams do not have to be fixated on a single technology. This is evident from the case studies of Uber.
- From the case study of Amazon [6] these are the following reasons as to why Amazon switched over to a microservice architecture -:
  - New thoughts and ideas could be effortlessly actualized, at a designer level and also in operational support.
  - Engineers got a more profound programming proprietorship and control. This helped with development and concentration on framework security and execution than was conceivable when all code lived in a monolithic application.
  - Scale their "operation independently, maintain unparalleled system availability, and introduce new services quickly without the need for massive reconfiguration.
- Better Fault Isolation: If one microservices fail, other will continue to work [7].
- Scalability and Reusability: Easy to scale and integrate with third-party applications.
- According to Netflix [8], these are the advantages that microservices provided them -:
  - Removal of friction from product development cycles increases the speed of the business as a whole.
  - Keep Code at Similar Level Of Maturity - If the new code has to be written create a new microservice for it and let the existing one work as well simultaneously. Deploy new features to the edited code and if everything works well merge the two codes. In this process, services are never stopped and constantly upgraded.
  - Speed and availability of the overall system are improved, while at the same time cost, size, and risk of software development to the business are reduced.
- Walmart successfully revitalized it's failing architecture with Microservices in the following ways -:
  - Walmart majorly incorporated microservices to improve scalability [5]. They could not handle 6 million page views per minute and user experience became really bad. With the use of Microservices these are the added advantages that Wal-Mart saw -:
  - Conversions were up 20%, Mobile orders increased by 98%, No downtime during Black Friday. Operational savings were significant. 40% of computing power was saved which involved 20-50% of cost saving.

7 WHY NOT USE MICROSERVICES?

Microservices is an architecture. It has its pool of advantages and disadvantages. The following are the downside to microservices.

- Developing distributed systems can be complex [9]. There can be a scenario where one of the services may not be responding, things can get more complicated when remote calls experience latency.

- Multiple databases and transaction management are difficult.
- Testing a microservices-based application can be cumbersome. But now, each dependent service needs to be confirmed before you can start testing.
- Deploying microservices can be complex. They may need coordination among multiple services, which may not be as straightforward as deploying a WAR in a container.

8 EXPERT OPINIONS

We specifically asked experts on Quora and Stack-Overflow their views regarding using microservices from day one. We got both good and bad opinions for it.

Below opinions backup Microservices from day one

1. *Puja Abbassi, working in the Container and Microservice space [10].*

"Markets are dynamic and when you start out you usually don't have product/market fit and you don't 100% know what the customer wants. Microservices enable you to be more agile and iterate faster on your product and actually innovate instead of working against your technical debt. If you have a good microservice architecture, it enables you to deploy several times a day, maybe even several times an hour if you're that fast. That speed and agility in your development process have a big influence on your business.

Breaking a monolith into Microservices later, has never worked out cleanly for me. Combining Microservices into a monolith is easy if you have the tooling."

2. *Dmitri Toubelis , Microservice Architect [11].*

"In startup business your first priority is ship your product fast. If you have qualified resources and you can develop microservices architecture from the get-go then do it by all means; it will pay back big time in the future. If you do not have qualified resources at your disposal then I wouldn't struggle about it too much - do what you know best."

3. *Dima Korolev, Principal Maintainer at C5T [12].*

"If you're an engineer or architect, by all means it's a win. Microservices are the new TDD. Eventually, they pay off in form of less technical debt, easier hiring, simplified ops, and faster scaling."

4. *Archis Gore, CTO at Polyverse [13]*

"I would only ever do Microservices. I've been doing them for years now, and am comfortable being around them. Breaking a monolith into Microservices later has never worked out cleanly for me.

Build microservices independently. Let them be built in a way that they are deployed independently and can be deployed independently at all times - like kernel modules."

The service owners can't cross boundaries when writing code, because they can't reference "common objects" unless they expose a service to allow that. But debuggers, testers, developers can still run the code as one giant debuggable monolith. And Ops/Infrastructure people can compose it on production clusters pretty well too."

5. *Martin Goodwell, Tech lead at dynatrace*

"Yes, absolutely. Service discovery, orchestration, and load-balancing are readily available. You won't need to build them yourself.Try to create services as small as possible when it comes to functionality. You'll benefit from much better maintainability (especially once new members join your dev team), scalability and resilience.

Below opinions backup monoliths -:

1. *Nicolae Marasoiu, SDE at Metro Group [14]*

"Be aware that microservices are suitable only in certain scenarios/use-cases, and perhaps it is best to split your software into multiple microservices only at a certain point, in which the boundaries of your domains have naturally stabilized".

2. *Rob Landers, Software Engineer at Automatic [15].*

If you have an excellent architect to figure out the contracts between each service, I'd say go with microservices from the outset. If not, go with monolith.Never do a microservice first, as you probably won't know what you're building (as a team) and what kind of bugs your users will find. Stay agile, and don't build a spaceship until you need to get to space.

3. *Ignacio Nicolás Rodríguez, Principal at Fast Microservices [16].*

"Do something outstanding. However it comes more fluidly from your mind into life. Think about architecture when there's something to put into one.

4. *Erik Hoffman, Solutions Architect [22]*

"Classic Monolithic has been a success and no point in moving to Microservices."

Insights from StackOverflow

Hans who has a reputation of 723 suggests:

"This is really opinion based, I would say it depends on your

team's experience.

Do you have previous experience building microservices? Are you able to quickly set up an infrastructure for things like service orchestration/discovery, service-to-service communication, auth/security, gateways? Or do you have experience with more managed solutions such as AWS Lambda/API Gateway, Google Cloud Function/Endpoints?

Other than the infrastructure, are you familiar enough with the domain of your startup to be able to precisely define service boundaries? Poorly defined service boundaries will definitely add unnecessary constraints to your software.

In my opinion, if you do not have prior experience with microservices, then start off with a monolith. Do your research on your domain, and keep your business logic clean and predictable, keep the interfaces of your monolithic code base as straight-forward as possible. Then later, you can start splitting parts of your monolith into other services. Provided that your code is clean, splitting parts of your codebase into another service is literally just moving that code to another deployment and instead of calling them directly, you call them through the network. You can repeat this process as you see fit to trim down the monolith over time. The "monolith" will become just another "service".

I think, unless you are well familiar with microservices, the additional time and resource invested in trial and error are not justified, since a startup is a fast-moving environment with very limited time and resource."

9 INTERVIEWS AND OPINIONS

We got a chance to Interview with Nithiwat Kampanya lead software developer at Virginia Cyber Range [16] and Aras "Russ" Memisyazici Network Architect at Virginia Cyber Range. They incorporated Microservices from day 1.

The Virginia Cyber Range is a Commonwealth of Virginia initiative with a mission to enhance cybersecurity education for students in the Commonwealth's public high schools, colleges, and universities. The Virginia Cyber Range seeks to increase the number of fully prepared students entering the cybersecurity workforce in operations, development, and research. They provide an extensive Courseware Repository for educators and a cloud-hosted Exercise Area environment for hands-on cybersecurity labs and exercises for students.

In this interview, they talk about their likes and dislikes for microservices and why did they start from day 1. Below are their answers.

1) Did you start with Microservices or Monoliths?

Nithiwat Kampanya: We started with microservices.

2) Why Microservices from day 1?

Aras Memisyazici: Nithiwat, I and the other people in our team are all very experienced and we could directly move to Microservices. Plus for cloud-related development and scalability Microservices is the key. We also had members with good expertise in different fields. We believed that dividing the system into well-defined services with clear interfaces would allow us to work more effectively on the project. We were also very confident that we will be needing more and more scaling. There will be more and more students and users in the coming future.

3) What is the most important benefit of microservices?

Nithiwat Kampanya: Quick changes in the system. Including a new route for API calls, making changes in exercises and assignments of students. Not worrying the changes would affect the entire application. The front-end developers are more free and same with the back-end developers as well. It has also helped in infrastructure automation, and automated code deployment and debugging.

4) Has Microservices helped you achieve your Goal?

Aras Memisyazici: We have done a great job so far. Our users are happy, we are happy. We provision systems to students and teachers where they can solve their exercises. We are using AWS for this. AWS has a good source integration for microservices. The cost for AWS is also comparably less while using microservices.

5) Will you recommend using Microservices to other startups or teams?

Aras Memisyazici: This again depends on various factors. If you have the right knowledge, the right target set in front of you then there is no point and fun in doing the wrong thing and then correcting it. If plans set straight, you should go for microservices. There were some scenarios wherein we started with monoliths at the start just for prototyping and seeing if the things actually work and we moved to microservices in the production stage.

6) Did models of any industry inspire you to use Microservices?

Nithiwat Kampanya: Not primarily. We knew our goals. But we did study models of Amazon, Netflix.

7) Challenges that you faced while building microservices.

Nithiwat Kampanya: Testing is a challenge with microservices. Testing interactions between services are challenging. Distributed systems are harder to program since remote calls are slow and are

always at risk of failure.
Managing developers who have primarily worked with Monoliths. It is a learning and new learning phase. Takes a bit of time.

8) How does Microservice help in agile development?
Aras Memisyazici: Microservices undoubtedly helps in scaling. Adding more people is easy. Let's the work go in parallel. We can also build more teams. Dependencies and coordination is a bit of challenge.

We also asked them the various techniques they have adopted from this. The major take away from this interview was, we got to know that because of their expertise they could directly move with microservices.

10 SURVEY ANALYSIS

We surveyed 20 Master students from Virginia Tech pursuing computer engineering and computer science. They are upcoming software developers and would soon be working in software industries. We interviewed with professionals and asked them questions, but we also wanted opinions on microservices from the upcoming and fresh minded developers.

Through this survey we wanted to know if the students knew about microservices, have they used them in projects before and how has it helped them. If they plan on using it in future projects, and if given a chance would they have started projects directly from microservices.
55% of the students had heard about microservices. If they did not know we explained them by providing a video link and also by mentioning a brief description. After this, we asked students if they felt Microservices is a mere hype. 90% believed microservices is not a hype and 10% believed otherwise.
We asked students if they have used microservices in any of their projects. 75 % did not use microservices before and 25 % of them had incorporated microservices in some form in their projects and applications.
For students who had not used microservices, 80% said they might be interested in using it, 10% believed yes they would be using it and the remaining 10% did not agree with it.
For students who had previously used microservices, 60% were developing it using frameworks, 10% using a platform and the remaining were using custom-built modules. We also asked them about the advantages offered by microservices. 80% of the students said the most significant advantage is technology independence where different languages could be used for different services. The remaining chose scalability as the major advantage.
From the 25% students who used microservices we asked if they would have started microservices from the very beginning to which we received 50% yes and 50% no response.
We were glad to know over 50% students had heard about microservices and do not think it as hype. The interest to learning microservices was also tremendous. Only a few worked with actual projects. If courses related to software engineering with microservice architecture are offered in college, it will largely benefit the students and make them industry ready. With the given knowledge it would open up a pathway to directly work on with Microservices from the very beginning and not utilizing resources later to make the monolith scalable.

11 CONCLUSION

From the opinions that we got, the major conclusion was if you have the expertise in your team you can go for microservices from the very start. If you think your application is going to scale huge, go with microservices from the very beginning.
From our research, we find the following points. If your organization is capable or has already maintained most of the following points then it can move onto microservices from day one. It will consume a lot of engineer's time and require great engineers as well. If the team is not experienced you can go with the monolith, build up quickly and then slowly shift to microservice or any other architecture if need be. Without knowing major of the techniques listed below monoliths can still be developed. Virginia Cyber Range was efficient in major of these and being a startup they could easily move onto microservices and are doing well.

- Basic Monitoring, instrumentation, health checks, Distributed logging and tracing.
- Ready to isolate not just code, but the whole build + test + package for every service.
- Can define upstream/downstream/compile-time/runtime dependencies clearly for each service
- Know how to build, expose and maintain good APIs and contracts.
- Know infrastructure automation this holds a very important key.
- Have or ready to invest in development tooling, shared libraries, internal artifact registries, etc.
- Good unit testing skills. When microservices are added the unit tests increases significantly and it also gets harder. The team should be ready for this because this work will increase exponentially. This will also bring rise to unit/contract/API tests and decrease End to End (e2e) tests [20].
- Aware of [micro] service vs modules vs libraries, distributed

monolith, coordinated releases, database-driven integration.

- Have engineering methodologies and process-tools to split down features and develop/track/release them across multiple services [20] (scrum, XP, waterfall, Jenkins etc.).
- Ready to honor compatibility, even if you're the same person consuming this service on the other side.
- Have working Continuous Integration(CI) and Continuous Delivery (CD) infrastructure

If an organization does not have the engineering robustness and maturity to effortlessly run a handful of services (12factor, CI, CD, integration, testing, etc), it will be a disaster to switch to many of them.

What we have observed is microservices cannot be learned as a framework or tool. It requires a holistic approach that comes with experience. You need creative and talented engineers with a very strong foundation in the whole software development lifecycle, from development to testing to deployment.

The microservices approach is just another tool. It can end up being a powerful business asset, or an unproductive developer bottleneck. It all comes down to how we leverage it.

REFERENCES

[1] Robert Blumen, Johannes Thönes- Microservices http://ieeexplore.ieee.org/document/7030212/

[2] A Balalaie, A Heydarnoori, P Jamshidi (2016), Microservices Architecture Enables DevOps: an Experience Report on Migration to a Cloud-Native Architecture, http://ieeexplore.ieee.org/document/7436659/

[3] Achieving Cloud Scalability with Microservices and DevOps in the Connected Car Domain, Dr. Tobias Schneider http://ceur-ws.org/Vol-1559/paper16.pdf

[4] Rewriting Uber Engineering: The Opportunities Microservices Provide https://eng.uber.com/building-tincup/

[5] Microservices Resource Guide https://martinfowler.com/microservices/

[6] I Love APIs 2015: Microservices at Amazon https://www.slideshare.net/apigee/i-love-apis-2015-microservices-at-amazon-54487258

[7] Benefits of Microservices Architecture Implementation https://dzone.com/articles/benefits-amp-examples-of-microservices-architectur

[8] Microservices at Netflix: Lessons for Architectural Design https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/

[9] Microservices architecture: advantages and drawbacks https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/

[10] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Puja-Abbassi?srid=n8hw

[11] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Dmitri-Toubelis?srid=n8hw

[12] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Dima-Korolev?srid=n8hw

[13] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Archis-Gore?srid=n8hw

[14] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Nicolae-Marasoiu?srid=n8hw

[15] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Rob-Landers-1?srid=n8hw

[16] https://virginiacyberrange.org/

[17] https://www.quora.com/Should-you-build-microservices-from-day-1/answer/Ignacio-Nicol%C3%A1s-Rodr%C3%ADguez?srid=n8hw

[18] Giovanni Toffetti, Sandro Brunner, Martin Blochlinger, Florian Dudouet, Andrew Edmonds Microservice Architecture Enables DevOps http://ieeexplore.ieee.org/document/7436659/

[19] Robert Blumen, Microservices http://ieeexplore.ieee.org/document/7030212/

[20] https://medium.freecodecamp.org/rest-in-peace-to-microservices-or-not-6d097b6c8279

[21] https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd

[22] https://www.quora.com/profile/Erik-Hoffman-3

.