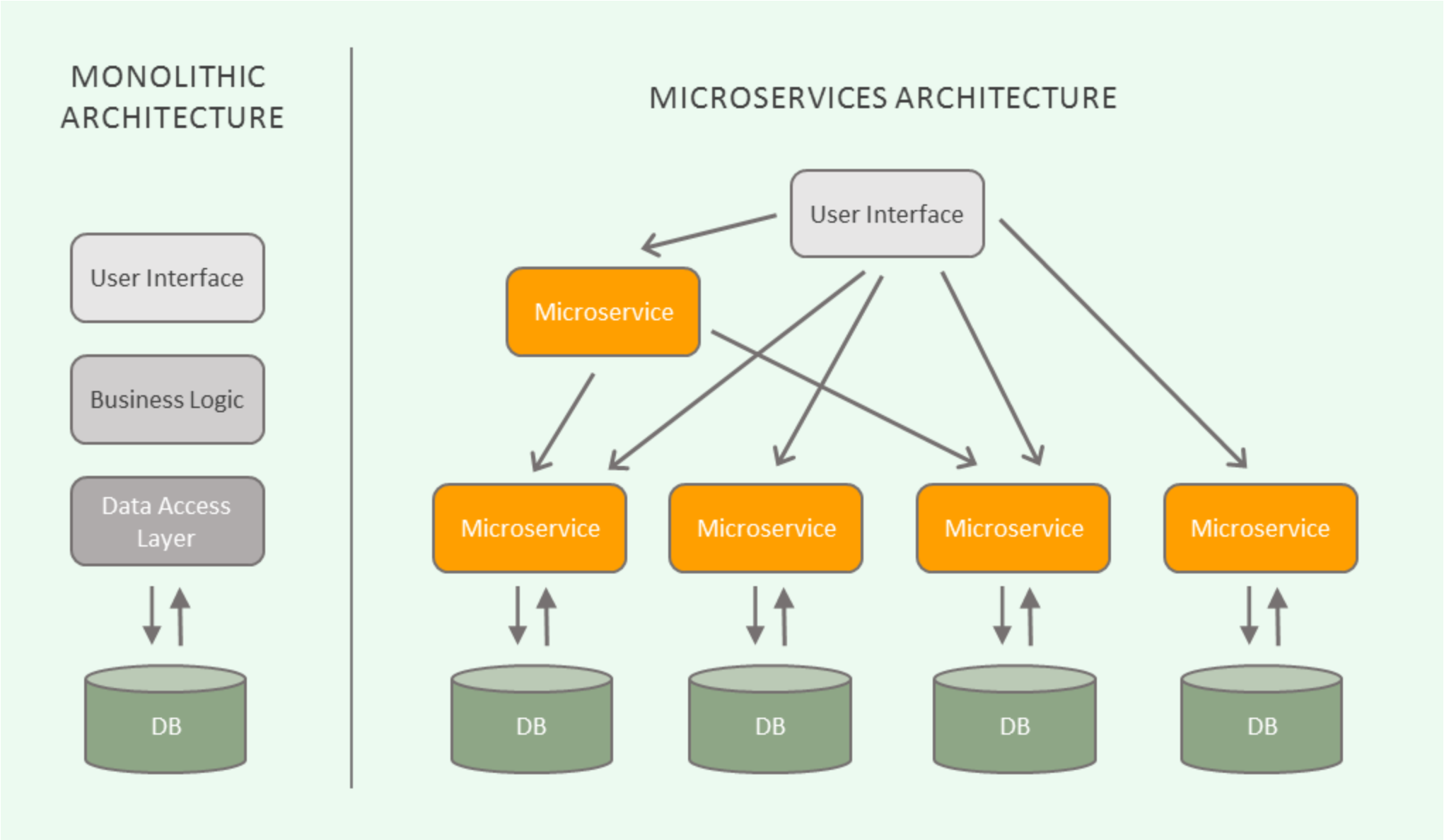
Introduction to Microservices: What are Microservices? Use Cases and Examples



Source: DZone <https://dzone.com/articles/what-are-microservices-actually>

A microservice architecture can beef up your team's speed by adjusting how they design and ship code, and developers and business leaders can get ahead by implementing it inside their teams. The one-two punch of serverless and microservices combined <https://algorithmia.com/blog/introduction-to-serverless-microservices/> is driving totally new types of applications and frameworks.

So what are microservices all about? The concept is based on a pretty simple idea: it sometimes makes sense to develop your applications as a lot of very small interlocking pieces instead of one giant whole. These components are developed and maintained separately from each other, so updates don't require re-doing the entire codebase. Along with a few other design requirements, that's the basic idea of microservices.

Understanding the "Traditional" Monolithic Model

Traditional application design is often called "monolithic" because the whole thing is developed in one piece. Even if the logic of the application is modular, it's deployed as one group, like a Java application as a JAR file for example. Imagine if all of your notes from different college classes were in one long stream.

This type of code writing and deploying is convenient because it all happens in one spot, but it incurs significant technical debt over time. That's because successful applications have a tendency of getting bigger and more complex as your company grows, and that makes it harder and harder to run.

For some insights into why and how a monolithic application can get confusing, consider this example from Chris Richardson <https://www.nginx.com/blog/introduction-to-microservices/>: "*I recently spoke to a developer who was writing a tool to analyze the dependencies between the thousands of JARs in their multi-million line of code (LOC) application. I'm sure it took the concerted effort of a large number of developers over many years to create such a beast.*"

There are a few reasons why this monolith eventually becomes so difficult to manage, including:

- The codebase is too big for any single developer to fully understand
- If the codebase is difficult to understand, changes made will often be detrimental
- Larger applications mean longer and longer deployment timeframes
- Agile frameworks often require multiple pushes to production each day, and re-deploying the entire monolith runs into time issues

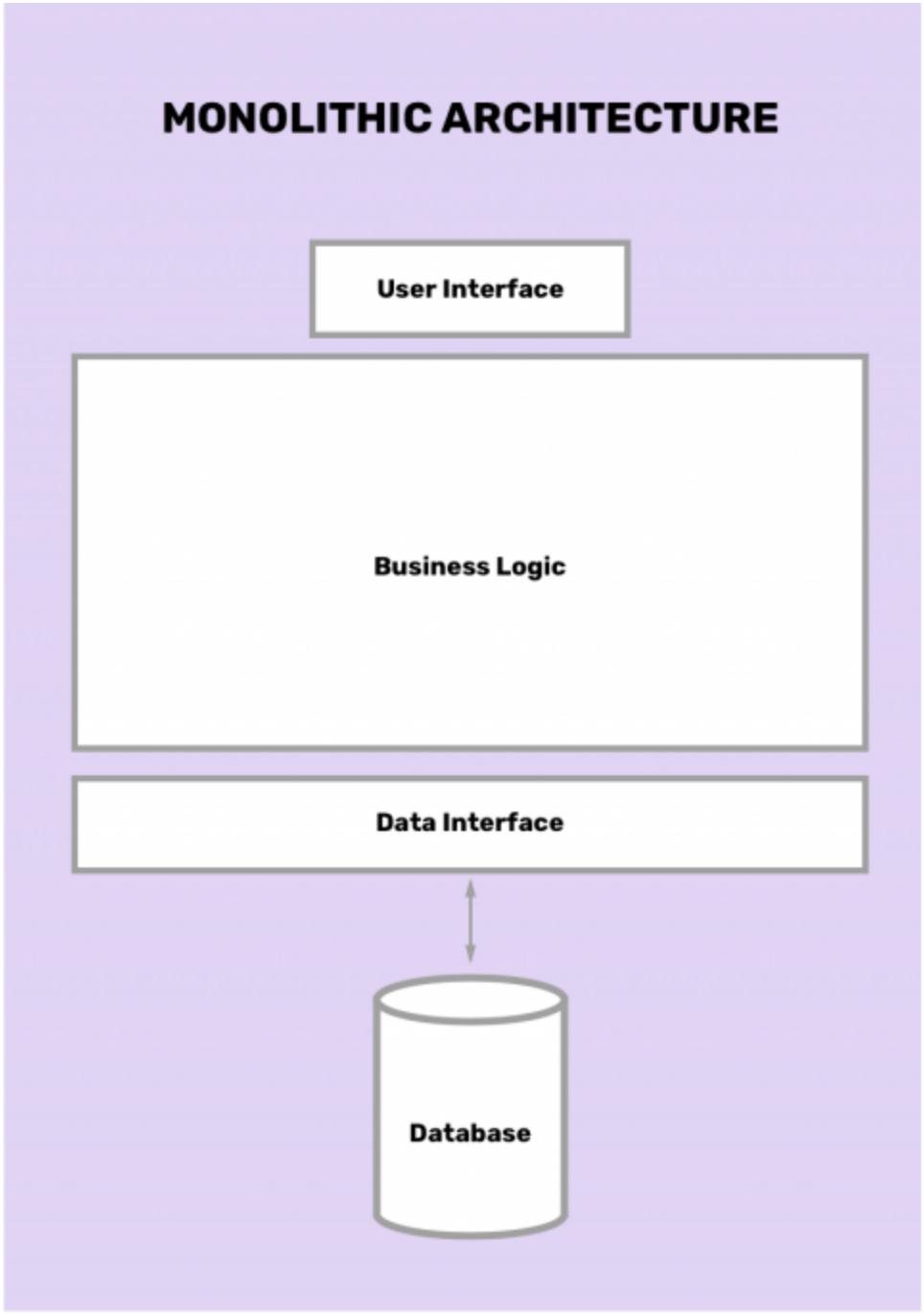
Because of these and many other accompanying issues, a new way of developing applications is becoming popular. Microservices separates all of the major parts of this monolith from each other, untangling the codebase and drastically changing how developers can write and interact with it.

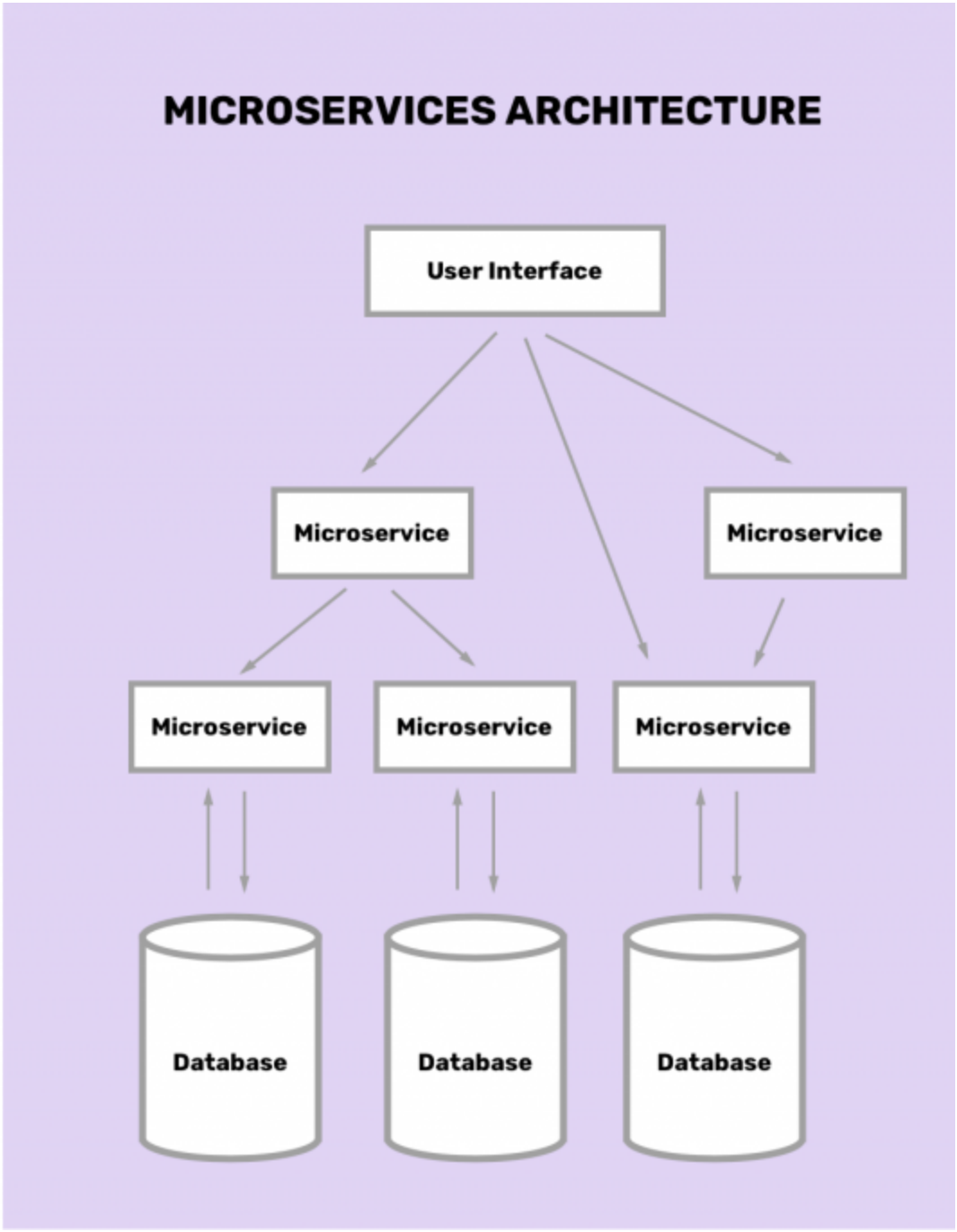
Service-Oriented Architecture (SOA) is used for applications composed of discrete and loosely connected agents that perform a function. SOA describes an application that can be built in such a way that its modules are seamlessly integrated and can therefore be easily reused. However, this type of architecture is very complex, sometimes sending over a million messages at a time, making it difficult to manage. SOA also often has higher response times and lower overall performance.

In contrast, microservices allows developers to build and manage software efficiently. It's not only easy to work with during the design and build phase, but also performs with speed and efficiency once launched. Monolithic and SOA architectures both require systematic changes to be made by modifying the monolith. Microservices remove this issue, along with many others, by untangling the monolith so that changes can be made by simply creating a new service.

What Are Microservices?

In contrast with the monolith type application, here's what an app developed with a microservices focus might look like:





Overall, it's largely the same: you have a user interface, some functions, and a database. With microservices though, those functions (not literally functions, but functional parts of an application) are all separate. They communicate with the user interface, each other, and instances of the database.

A team designing a microservices architecture for their application will split all of the major functions of an application into independent services. Each independent service is usually packaged as an API so it can interact with the rest of the application elements.

In direct consonance with the problems outlined above, breaking down your application into bundles of microservices offers some key benefits:

- Simplify your application with well-defined boundaries for each piece of functionality
- Allow teams to work separately on independent parts of your application without the need for constant collaboration
- Microservices can be deployed, maintained, updated, and scaled independently of each other in a continuous fashion

It's hard to overstate how large of a paradigm shift this new approach is. It totally flips around many of the challenges of traditional monolith deployment.

Microservices Examples and Business Use Cases

What is the use of microservices? A microservices-based architecture offers a lot of benefits in theory, but it's difficult to make it work in practice. That's why we're still very much in the early development stages of this idea, and that applies even more strongly to larger companies.

But this isn't entirely now. In fact, the concept of splitting applications into smaller interactive parts has actually been around as a programming paradigm for a while. One of the reasons why it's taken this long for microservices to emerge as a legitimate alternative is simple: culture. Implementing this architecture isn't just a technical decision: it's about having the right teams in place, being comfortable using open source, and working in an organization that's comfortable challenging the status quo in IT.

Companies implementing microservices have been very open about their process and why they chose it. Here are some useful examples from companies that might not surprise you:

- Service-Oriented Architecture: Scaling the Uber Engineering Codebase As We Grow <https://eng.uber.com/soa/> (Uber)
- Netflix Conductor: A <https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40>M <https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40>icroservices <https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40>O <https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40>rchestrator <https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40> (Netflix)
- What Led Amazon to its Own Microservices Architecture <https://thenewstack.io/led-amazon-microservices-architecture/> (Amazon)

But in addition to the usual large-tech-company repeat offenders, some companies that are utilizing this architecture might surprise you:

- Partial Failures in a Microservices <https://qconnewyork.com/ny2015/ny2015/presentation/partial-failures-microservices-jungle-survival-tips-comcast.html>Jungle: <https://qconnewyork.com/ny2015/ny2015/presentation/partial-failures-microservices-jungle-survival-tips-comcast.html>Survival Tips from Comcast <https://qconnewyork.com/ny2015/ny2015/presentation/partial-failures-microservices-jungle-survival-tips-comcast.html> (Comcast)
- The eBay Architecture <http://www.addsimplicity.com/downloads/ebaysdforum2006-11-29.pdf> (eBay)
- Walmart Embraces Microservices to Get More Agile <http://www.baselinemag.com/enterprise-apps/walmart-embraces-microservices-to-get-more-agile.html> (Walmart)

IT organizations are still figuring if they're willing to make this shift. But in the meanwhile, those who do and find the right fit are reaping the benefits.

Challenges with Deploying Microservices

As with any design decision, there are drawbacks to a microservices architecture. The major issue is complexity––breaking up your codebase makes it easier to understand, but creates complications in orchestration. Microservices mean a distributed system, which comes with its own problems.

Your team is going to need to handle some new situations. For example, any individual microservice can fail at any point, just like a traditional software deployment. You need to write logic to deal with that. Another issue is database management––with the monolith there's typically only one or a few databases to update, but with microservices there can be many. Managing data consistency across a distributed system can be a major challenge.

Finally, testing and deployment can become troublesome in a microservices-oriented architecture. If any services are dependent on others, you need to design a specific order for deployment and testing. Changes can impact multiple services in your application, and accounting for that is difficult.

Serverless, Microservices, and Containers

The shift towards microservices fits nicely with two other important trends in the deployment space: serverless and containers. Serverless is about abstracting the code around server side logic, and having a provider manage your infrastructure for you. We wrote about it more in-depth here <https://algorithmia.com/blog/serverless-computing/>. Containers are all about bundling your code and dependencies into self-executing, independent packages.

Containers and microservices fit together because they have the same fundamental goal—package individual components as independent, responsive elements. Serverless empowers this architecture by focusing on functions as a service––now that your application pieces are packaged individually, deploying them as functions can make a lot of sense.

Algorithmia's architecture brings all of these elements together by offering an easy-to-use platform for serverless deployment of algorithms as microservices <https://algorithmia.com/>. You get all the benefits of a microservices architecture, but it's simple to orchestrate and integrate. Algorithmia is also the only serverless platform that offers GPUs, which are a key part of building a fast machine learning application.

Why Microservices Are Killer for Machine Learning

As more machine learning goes into production <https://algorithmia.com/blog/deploying-machine-learning-at-scale/>, it's becoming clearer that a microservices architecture can be a good fit for this kind of application. There are two major reasons why this is the case:

- After training your models, inference is usually stateless—since no data or state needs to be maintained, independent services work
- Machine learning is a compute-intensive process that often requires specialized hardware (like GPUs), and you don't want that to be a core part of your server requirements

Algorithmia deploys algorithms as scalable microservices to take advantage of these two features.

Microservices Resources

Further Reading and Papers

Introduction to Microservices <https://www.nginx.com/blog/introduction-to-microservices/> (Nginx) – "*This blog post is the first in a seven-part series about designing, building, and deploying microservices. You will learn about the approach and how it compares to the more traditional Monolithic Architecture pattern. This series will describe the various elements of a microservices architecture. You will learn about the benefits and drawbacks of the Microservices Architecture pattern, whether it makes sense for your project, and how to apply it.*"

Introduction to Serverless Microservices <https://algorithmia.com/blog/introduction-to-serverless-microservices/> (Algorithmia) – "*With the rise of AI / Machine Learning, microservices are more important than ever. As teams adopt microservice-oriented architectures, often serving powerful ML models, they build better products faster, outpacing their competition.*"

Microservices <https://martinfowler.com/articles/microservices.html> (Martin Fowler) – "*The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.*"

Architecting Microservices <http://ieeexplore.ieee.org/document/7958492/> (Paper) – "*This paper reports on a PhD research project addressing three different challenges concerning MSA: (i) the identification of the key properties of microservice architectures, (ii) the identification and investigation on a description language for designing and analyzing architectures, (iii) the identification of the factors that impact the process of migrating existing applications towards MSA. The initial contributions of this project are: (i) a systematic mapping study on architecting microservices performed in order to understand the state of the research and the possible gaps in the area, (ii) an approach for architecture recovery of microservice-based systems named MicroART, and (iii) the implementation of the MicroART first prototype.*"

Microservices: Yesterday, Today, and Tomorrow <https://arxiv.org/abs/1606.04036> (Paper) – "*Microservices is an architectural style inspired by service-oriented computing that has recently started gaining popularity. Before presenting the current state-of-the-art in the field, this chapter reviews the history of software architecture, the reasons that led to the diffusion of objects and services first, and microservices later. Finally, open problems and future challenges are introduced. This survey primarily addresses newcomers to the discipline, while offering an academic viewpoint on the topic. In addition, we investigate some practical issues and point out some potential solutions.*"

Microservices Tutorials and Walkthroughs

Microservice Architecture Tutorial <https://www.tutorialspoint.com/microservice_architecture/index.htm> (tutorialspoint) – "*Microservice Architecture is a special design pattern of Service-oriented Architecture. It is an open source methodology. In this type of service architecture, all the processes will communicate with each other with the smallest granularity to implement a big system or service. This tutorial discusses the basic functionalities of Microservice Architecture along with relevant examples for easy understanding.*"

Spring Boot Tutorial: REST Services And Microservices <https://jaxenter.com/spring-boot-tutorial-rest-services-and-microservices-135148.html> (Jaxenter) – "*The times of Java EE application server and monolithic software architectures are nearly gone. Hardware is not getting faster anymore, but internet traffic is still increasing. Platforms have to support scaling out. Load must be distributed to several hosts. Microservice-based architectures can offer solutions for this requirement. Apart from the better scaling, microservices offer faster development cycles, dynamic scaling depending on load and improved failover behavior.*"

Quick Intro to Node.JS Microservices: Seneca.JS <https://www.codementor.io/ivan.jovanovic/introduction-to-nodejs-microservices-senecajs-du1088h3k> (Codementor) – "*So, you want to use NodeJS to create microservices architecture? That's very simple and awesome! In my career, I've used many frameworks and libraries for creating microservices architecture, even created custom libraries (don't do it!) — until I found SenecaJS* <http://senecajs.org/>."

Build An API For Microservices In 5 Minutes <https://www.javaworld.com/article/2952103/enterprise-java/build-an-api-for-microservices-in-5-minutes.html> (Javaworld) – "*Enough talk, let's roll up our sleeves and start building our microservices core competency. This brief, hands-on tutorial shows you how to create a new API with AnyPresence JustAPIs. Before you start,download the free trial version of JustAPIs and follow the Quick Start Guide to set it up.*"

Books

Python Microservices Development (Ziade) – "*We often deploy our web applications into the cloud, and our code needs to interact with many third-party services. An efficient way to build applications to do this is through microservices architecture. But, in practice, it's hard to get this right due to the complexity of all the pieces interacting with each other. This book will teach you how to overcome these issues and craft applications that are built as small standard units, using all the proven best practices and avoiding the usual traps.*"

Building Microservices with .NET Core 2.0 (Aroraa) – "*Moving forward, you will be introduced to real-life application scenarios; after assessing the current issues, we will begin the journey of transforming this application by splitting it into a suite of microservices using C# 7.0 with .NET Core 2.0. You will identify service boundaries, split the application into multiple microservices, and define service contracts. You will find out how to configure, deploy, and monitor microservices, and configure scaling to allow the application to quickly adapt to increased demand in the future.*"

Building Microservices: Designing Fine-Grained Systems (O'Reilly) – "*Distributed systems have become more fine-grained in the past 10 years, shifting from code-heavy monolithic applications to smaller, self-contained microservices. But developing these systems brings its own set of headaches. With lots of examples and practical advice, this book takes a holistic view of the topics that system architects and administrators must consider when building, managing, and evolving microservice architectures.*"

Microservice Architecture: Aligning Principles, Practices, and Culture (O'Reilly) – "*Microservices can have a positive impact on your enterprise—just ask Amazon and Netflix—but you can fall into many traps if you don't approach them in the right way. This practical guide covers the entire microservices landscape, including the principles, technologies, and methodologies of this unique, modular style of system building. You'll learn about the experiences of organizations around the globe that have successfully adopted microservices.*"