Compunnel Digital

THE MICROSERVICES HANDBOOK

CONTENTS

THE MICROSERVICES HANDBOOK 03
WHAT IS MICROSERVICES ARCHITECTURE? 04
MICROSERVICES VS MONOLITHIC ARCHITECTURE 05
KEY FEATURES OF MICROSERVICES 07
HOW MICROSERVICES WORK 12
BENEFITS OF MICROSERVICES 13
DRAWBACKS OF MICROSERVICES 14
WHEN TO SWITCH TO MICROSERVICES 15
CASE STUDY - AVITRU 17
ABOUT COMPUNNEL DIGITAL 19

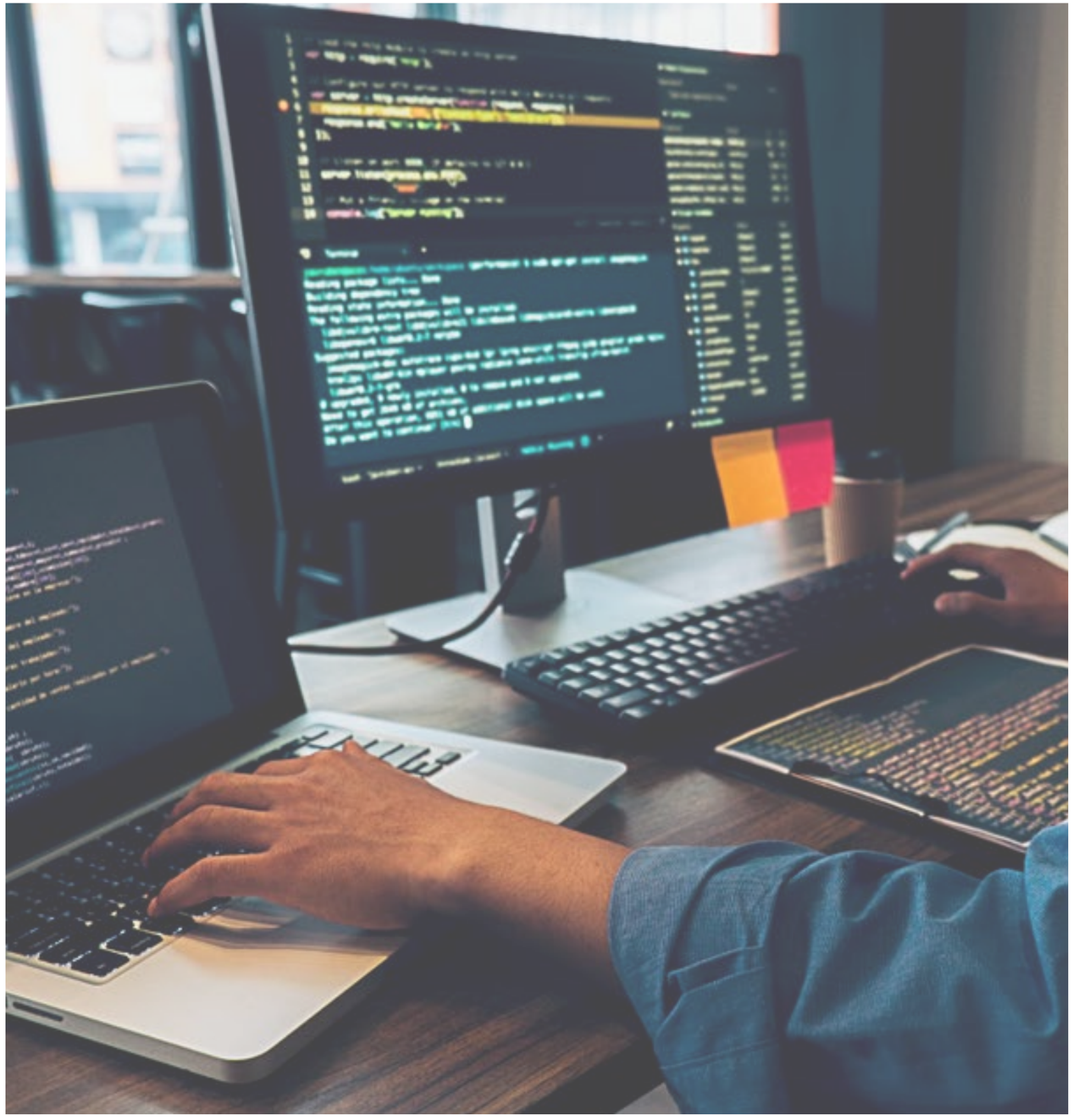
The Microservices Handbook

In today's fast paced business environment, companies must be able to quickly evolve to keep up with ever-changing technologies and customer demands. Microservices architecture allows organizations to rapidly respond to changes in business needs, laying the foundation for ongoing innovation and agility. With tech giants like Netflix and Amazon using microservices to rapidly scale and achieve continuous integration and continuous delivery (CI/CD), demand for the architectural style is higher than ever before. A 2018 survey of app developers showed that 36% currently use microservices, and another 30% are considering them in the future.

The hype is real, but how do you know if microservices are the right choice for your business? This eBook outlines the details, functionality, benefits, and drawbacks of using microservices architecture and when the right time is for your organization to take the leap.

What is Microservices Architecture?

Microservices architecture evolved out of Service Oriented Architecture. It is a style that structures an application as a collection of smaller services that are loosely coupled, independently deployable, and easily testable. In contrast to a monolithic architecture where the entire application is developed in one piece, each microservice is continuously developed and separately maintained. The services are autonomous and can be written in different languages and can even have different data storage techniques. API's (application programming interfaces) facilitate communication between the services and allow them to function in tandem to form one larger application.

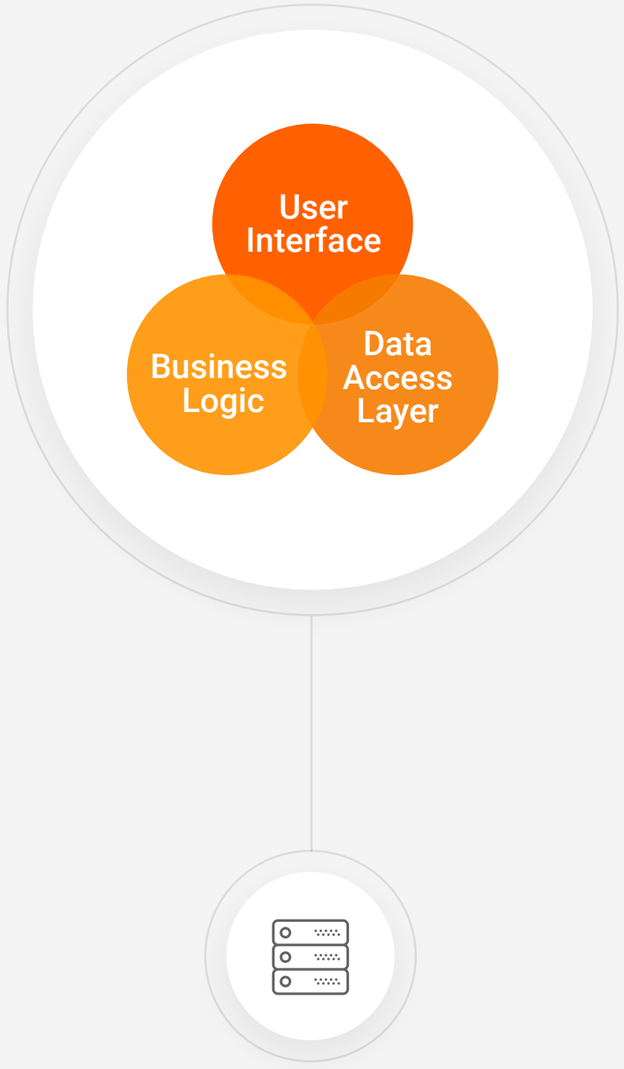


Microservices vs Monolithic Architecture

Monolithic:

In a traditional monolithic architecture, an application is developed by one team with one database and code set. The simplicity of having all modules and functionalities housed in one application makes it easier to develop and launch a product quickly, but problems can arise when trying to push updates or scale. The entire application has to be taken offline to be updated, meaning a lag in service time and a higher risk that any unchecked bugs could take down the whole program. Scaling can also be challenging, as the product can become too large and bogged down as it grows. Challenges aside, however, a monolith is the most cost-effective way to build a new application, and it can be easily managed by a smaller team.

Pros	Cons
• Easy for a smaller team to manage • Same code base throughout • Lower overhead costs • Quick go-to-market	• Difficult to test once launched • Difficult to scale • Updates must be done to entire app • Inflexible code set

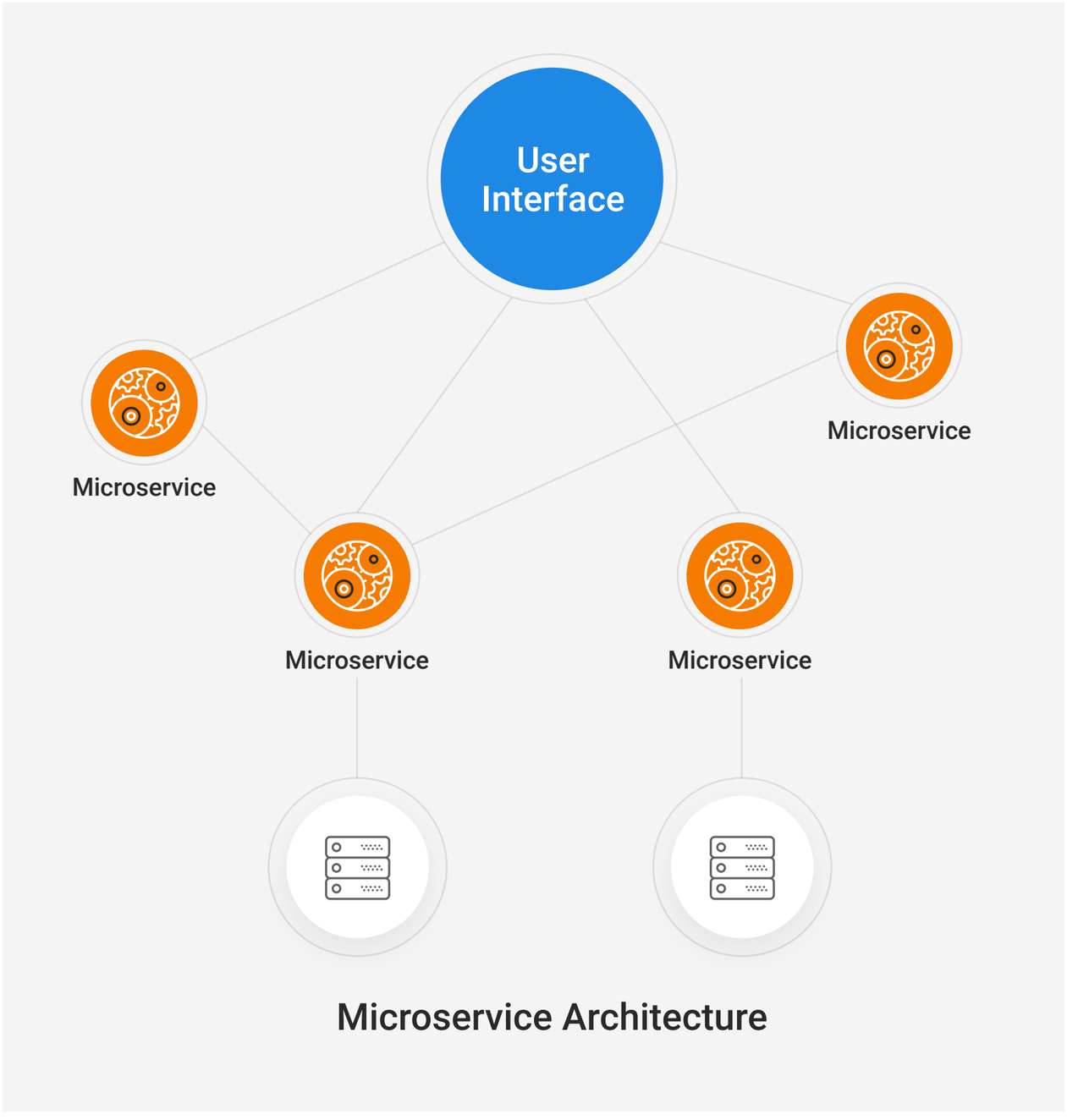


Monolithic Architecture

Microservices vs Monolithic Architecture

Microservice:

The microservice architectural style is way of building a single application as a suite of small services. Each one runs its own process and communicates with other services through API's. Because each service operates separately from the others, each one can be written in the codebase most conducive to its task. The services in microservice architecture are all independently deployable and written to address specific business needs, which makes testing and updates painless, as services can be taken offline one at a time without disrupting service to the application at large. Scaling is also much easier, as decoupled services give teams the freedom to ramp up certain segments of the application as needed. A good example of this is the shopping cart service during the holidays: it can be scaled up or down depending on seasonal demand.



Microservice Architecture

Key Features of Microservices

- Independent modules
- Communication
- Storage
- Maintenance
- Containers

Independent modules

The philosophy behind microservices architecture is that an application is easier to build and maintain when it is broken down into smaller parts that work together seamlessly. Each functionality of the application is broken down into independent modules that are responsible for performing precisely defined tasks.

Key Features of Microservices

- Independent modules
- Communication
- Storage
- Maintenance
- Containers

Communication

The individual services communicate with each other through a series of simple and universally accessible API's. Even if each module is written in a different code base, API's allow them to work together as one.

Key Features of Microservices

- Independent modules
- Communication
- Storage
- Maintenance
- Containers

Storage

Microservices don’t have to use the RAM and CPU resources of a traditional architecture, and each service can be connected to its own database or database schema, allowing for horizontal scaling.

Key Features of Microservices

- Independent modules
- Communication
- Storage
- Maintenance
- Containers

Maintenance

Each service is independently maintained by its own team within a software development organization. This separation of responsibilities allows for specialization and easy scalability of complex applications, as work can be done on individual services without disrupting other teams or the application at large.

Key Features of Microservices

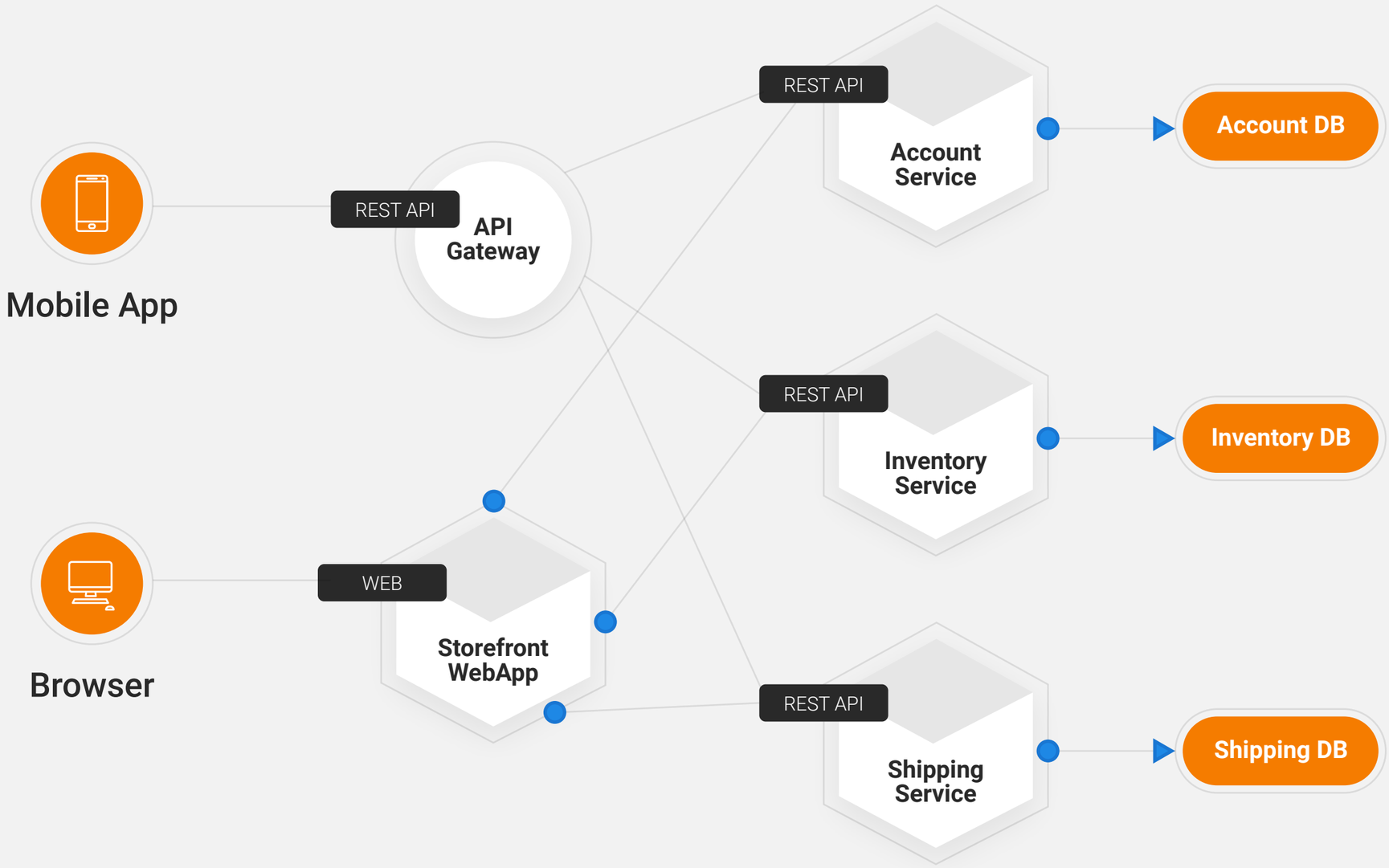
- Independent modules
- Communication
- Storage
- Maintenance
- Containers

Containers

Each service is often placed within a container that holds the libraries and executables needed for its specific tasks. Although each container operates independently, it still answers to directives from the main orchestration tool.

How Microservices Work

In this e-commerce example, each service is containerized with its own database solution, and all of them communicate with each other and the user interface through API's.



Benefits of Microservices

Organized Around Business Capabilities

Microservices allows teams to focus on business capabilities and not only technologies. Services are adaptable for use in multiple contexts, and each one can be reused in more than one business process depending on the need.

Increased Resiliency

Because the services are independent and self-contained, when one fails it does not impact the functionality of the others or the application at large. Separate services mean that fault isolation is swift and easy.

Faster Time to Market

Updates and changes can be made to individual services as required, without the need to re-write the entire codebase of the application.

Improved Scalability

Business-critical services can be easily scaled up and down without impacting the performance of the other services.

Code Flexibility

Each service can be written in the code base that is best suited to its specific task. This also allows develop to tailor the way information is presented to different audiences.

Ease of Outsourcing

Non-business critical services can easily be outsourced to third parties without risking the loss of intellectual property.

Drawbacks of Microservices

Complexity

The larger the application becomes the more complex the network of moving parts is. This can be difficult to manage for small or inexperienced teams.

Requires a Cultural Shift

For microservices architecture to work, teams need to be cross-functional and embrace the notion that each member is responsible for both successful production and deployment. A DevOps culture must be implemented.

Global Testing is a Challenge

Because the components are separated, spinning up test environments of the entire application can be complicated and challenging.

High Initial Investment

Building an application using microservices architecture requires a large up-front investment in hardware and software engineers. Although ROI generally improves once the application is functional, the cost of starting with microservices makes it prohibitive to some smaller organizations.

When to Switch to Microservices

1

You want to remove dependencies

The philosophy behind microservice architecture is for each service to be highly specialized and completely autonomous, making it independently deployable and easy to replace. If one service fails it can be isolated and repaired without taking down the entire system. This also makes testing a breeze, as you can pull one service at a time to be tested without disrupting the functionality of your larger product.

2

Your product is highly dynamic

Customers expect the products that they interact with to be ever-evolving: constantly keeping up with trends in innovation to enhance the user experience. If you are in a dynamic industry or have a highly dynamic product, microservices architecture gives your team the freedom to constantly evolve and launch new releases without disrupting service. This ability to provide continuous delivery and deployment allows you to wow customers with your agility, while keeping product development costs low.

3

Scalability is a priority

Microservices is a particularly attractive architecture if you need to enable support for a range of platforms and devices. Each independent service can be easily scaled to respond to increases in demand or usage. For example, an e-commerce business might consider migrating to microservices architecture in order to meet demand and mitigate costs during the holidays. If the shopping cart is an independent service, it can be scaled up to accommodate the extra traffic and prevent service disruptions for customers. This also has the added benefit of enabling “pay as you go” server usage, which leads to huge cost savings.

When to Switch to Microservices

4

You want the freedom of Cloud-based services

The benefits of migrating to the cloud are well known, and chief among them is cost efficiency. In a microservice environment, any number of independent services can be outsourced to the cloud or 3rd party platforms. Moving services to the cloud saves you from investing in on-site servers, software and staff, and with most cloud companies offering a subscription-based service, you pay only for the data you use.

5

You have an established product

It’s important to realize that microservices architecture is not a one-size-fits-all solution. Making the transition requires a substantial investment of time and money, and it is best suited to companies with an established, proven product. Building an MSA system can lengthen product to market time, making it a risky choice for start-ups or new, untested products. Although there is compelling argument to build your product using microservices from the ground up, it’s important to ask yourself: do you want a product that can scale but doesn’t work, or a product that works but can’t scale?

Case Study - AVITRU

Current State:

Technology	Business
• Sole dependency on a single, unsupported desktop application. • No support for real-time online and mobile co-editing tools. • Missing P2P virtual collaboration. • No content based search for Azure Database	• Inability to support an increasing number of customers using mobile devices. • Lacked the omnipresence and versatility

Future State:

- Collaborative digital platform for users to support parallel operations.
- Reuse of styling implemented on previous documents using document parser.
- Commercialized access to data for data oriented customers using Microservices.
- Microservices implementation in Azure Database to provide seamless information extraction from a pool of documents.

Avitru leverages a legacy desktop application with no mobile support, hence hampering a large customer pool using mobile to do their jobs.
High volume documents were difficult to manage within the application. Client wanted to make global appearance and easy maintainable system and was looking for a solution for its Azure Database to manage content for its upcoming & existing documents.

Construction Industry

Case Study - AVITRU

Transformation Summary:

Compunnel Digital transformed and developed a cloud-based digital platform for the client enabling users to collaboratively create, edit and manage specifications documents 24x7 and across different online and mobile devices.

Compunnel also developed a digital solution for extracting information based upon search keywords from Azure database using Microservices.

Key feature of our solution includes:

- Omni-channel integration platform for seamless user experience across all touchpoints –web and desktop using Microservices.
- Smart management of digital information by creating document parser.
- Commercialized access to data for data oriented customers using Microservices.

Achieved Outcome:

- Cloud-based digital platform with zero downtime enabling uninterrupted customer access 24*7
- 35-40% reduction in time to create new construction projects.
- 15-20% reduction in time to complete existing construction projects
- 45-55% enhancement in system availability
- 20-25% improvement in online user collaboration

Avitru leverages a legacy desktop application with no mobile support, hence hampering a large customer pool using mobile to do their jobs.
High volume documents were difficult to manage within the application. Client wanted to make global appearance and easy maintainable system and was looking for a solution for its Azure Database to manage content for its upcoming & existing documents.

Construction Industry

Compunnel Digital

For more than 25 years, we have strategically leveraged digital technologies to deliver competitive advantage to our clients

DIGITAL TRANSFORMATION IS A JOURNEY
Take 10 minutes to understand where you are before you make the next investment.

Our Digital Transformation Readiness Assessment delivers you a snapshot of the current state of your business so you can prioritize your digital initiatives.

Compunnel Digital

11 Broadway, Suite 632, New York, NY 10004
info@compunneldigital.com
www.compunneldigital.com

CompunnelDigi
609-606-9009