MICROSERVICES: A CHEAT SHEET

TechRepublic.

TABLE OF CONTENTS

MICROSERVICES: A CHEAT SHEET

If you want to build software that is more resilient, agile, and faster, you can do just that with microservice architecture.

BY BRANDON VIGLIAROLO

If you are involved in the software development world, even indirectly, you've probably heard someone mention "microservices." But just what does that mean?

In the past, applications were developed using monolithic architecture, meaning that all the dependencies, subroutines, libraries, and necessary snippets of code had to be built into the app directly. If any one of those elements goes bad, the entire application can come crashing to a halt.

Software monoliths have other problems, too: A simple update can leave an IT team with hours of work reconnecting data sources, making sure clients are updated, or addressing unforeseen problems that a small bit of new code causes in an unexpected place.

In short, software development has long been plagued by the need to program everything in one place using a style of software architecture that doesn't meet modern cloud-based computing needs. But what's the alternative? It's microservices architecture.

WHAT ARE MICROSERVICES?

Microservice architecture is complicated and many of the articles and blog posts outlining it from companies like IBM make the argument that a single definition of microservices is difficult to come by.

One of the most satisfying definitions I've seen comes from software developer Martin Fowler, who describes microservice architecture as "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource application programming interface (API)."

Microservices are especially useful as more and more computing moves into the cloud, which makes maintaining monolithic applications difficult and old-fashioned. Why put all of your eggs in a single cloud basket that can limit your ability to scale, make changes, and reduce stability?

Fowler points out that monolithic applications require complete duplication onto new servers to scale, eating up a lot of computing resources. Microservice architecture, on the other hand, allows small, independent components to be deployed to different servers, where they can be scaled or replicated as needed.

IBM further explains how microservices work, and defines it as a cloud-native approach to building applications from “loosely coupled and independently deployable smaller components, or services” that:

- Have their own stack;
- Use REST APIs and other forms of communication to connect to other services; and
- Are sorted by business capability and separated into “need-to-know” chunks via bounded context.

Some of the advantages of opting for a microservices approach over a monolithic one can be assumed from its contrast to monolithic design, but it’s still worth stating them:

- Because they function independently, each component can be built in whatever language the developer of that component prefers.
- Individual components can be updated on their own since their internal operation is inconsequential to the app as a whole. As long as they report back the correct data, the app will continue working.
- Scaling can be done to each individual component, saving time and computing resources spent replicating entire applications to account for load on a single component.
- A component no longer meeting the need of its application can be replaced without affecting the app as a whole.
- Applications can be decentralized between multiple cloud providers, servers, and other services both local and remote.

To summarize, microservices are an app design architecture with a philosophy based on building independent components that all connect via API to reduce complexity, increase scalability, and allow applications to be distributed with more ease than traditional monolithic-style program architecture.

HOW ARE MICROSERVICES DIFFERENT FROM SERVICE-ORIENTED ARCHITECTURE?

Comparisons are frequently made between microservices and service-oriented architecture (SOA). While the two may seem similar at first glance, they’re nearly completely different except in the most basic ways.

Both SOA and microservices involve the creation of small components that communicate data to other services, but the scope, purpose, and how the communication occurs are completely different.

For starters, SOA is an enterprise-wide architecture, whereas microservice architecture is a way to build a single application. The idea behind SOA is to create a common framework for communication that allows applications, data sources, and other network-connected elements to communicate in a platform-agnostic manner.

SOA wants communication between elements to happen fast, smooth, and without barriers; this is a radical difference from microservices, which want independent elements that aren't dependent on each other at all. SOA integrations are reused constantly—that's the goal of SOA, according to IBM. In the case of microservices, reuse is completely undesirable--if a component is being called in more than one place by its main application, agility and resilience will suffer.

Microservice architecture is all about independence, swappable parts, and speed. SOA, on the other hand, is designed to create network synchronicity, which is a completely different goal.

WHAT ARE USE CASES FOR MICROSERVICES?

There's a lot to think about when it comes to microservices and not all of it is easy to wrap your head around. One of those difficult elements is seeing the use cases for microservices which, while plenty, can be difficult to identify if you're not well-versed on the particulars of how microservices work.

Software consulting firm QAT Global points out six potential use cases for microservices:

- **Modernizing legacy applications:** Legacy software can be attached to individual microservices to make those systems accessible to newer software. When doing so, plan a microservice around a specific business functionality.
- **Big data:** Microservices are a natural fit for big data applications and can fit seamlessly into a data pipeline to perform specific tasks. Examples include identifying fraud, managing traffic, examining medical research, etc.
- **Real-time processing**: The publish-subscribe framework behind microservices can be used to process data in real time and deliver immediate outputs for banking, customer service, data streaming, and more.
- **Simplifying growth**: Apps and organizations that are likely to grow rapidly can make use of microservices because of how easily they scale. In addition, individual modules can be re-deployed for various purposes, reducing development time and speeding time-to-market.
- **Advanced analytics environments**: Machine learning, manufacturing, and the energy sector can all use microservices to validate computational models, perform A/B testing, and multivariate testing.
- **Distributing resource-intensive tasks**: Computing that eats up a lot of RAM or CPU can be distributed to multiple microservices across multiple machines, making it easier to perform those tasks, redeploy resource-intense services, and reduce load time.

These six use cases cover much of what microservices can be used for, but not all. Microservices may have been around for several years, but they're still an emerging technology that has many applications yet to be discovered.

ARE MICROSERVICES A MATURE METHOD FOR BUILDING APPLICATIONS?

The concept of microservices has been around since at least 2005, when Peter Rodgers gave a talk about micro-web-services at the Cloud Computing Expo. Since then, the microservices market has exploded. A report from Verified Market Research even expects it to hit $3.1 billion in value by 2026, a growth rate of more than 20% over its value in 2019.

Whether or not microservices are mature is likely an individual opinion, but the tech industry seems to think so--at least based on adoption rates. TechRepublic Premium's research seems to back this up, finding that most organizations are familiar with microservices and 73% have already integrated them into their development processes.

Companies that have begun using microservices reported to TechRepublic that they've seen benefits like faster deployments, greater flexibility in the face of changing conditions, and quick scaling, all of which are to be expected, given what microservices are designed to do.

Those still wondering if microservices are mature enough for their organization need to think of them less as a product and more as a design philosophy. The framework for microservices is there in the current tech world and additional tools aren't needed. Microservice architecture is simply a new way of thinking about designing software that puts the future of applications more in line with the cloud-based, decentralized world of computing that is increasingly becoming the norm.

CAN EXISTING APPLICATIONS BE TRANSFORMED INTO MICROSERVICE-BASED ONES?

The short answer to whether or not existing monolithic applications can be transformed into microservice ones is yes, but technology consultant Zhamak Dehghani said it's "an epic journey" that presents a lot of challenges.

Going down the path of deconstructing a monolithic application into microservices, Dehghani said, "requires on demand access to deployment environment, building new kinds of continuous delivery pipelines to independently build, test, and deploy executable services, and the ability to secure, debug, and monitor a distributed architecture."

Dehghani suggests that the first step should be to decouple a simple service, build out underlying infrastructure, and then work on moving large, vertical services that don't change very often.

One thing to definitely avoid, Dehghani said, is the temptation to extract existing code into an independent service, which she said often "causes the developers, and more importantly technical managers, to disregard the high cost and low value of extracting and reusing the code."

Instead, don't be afraid to revisit certain modules of code and see if they're current with organizational needs and objectives. It's entirely possible some new code could do a better job, but that means committing to some major development initiatives.

No matter how you look at it, breaking a monolithic application down into microservices will take a lot of work, requires extensive planning, and will probably involve spending a lot of money in order to save more in the future.

HOW CAN BUSINESSES AND DEVELOPERS GET STARTED USING MICROSERVICE ARCHITECTURE?

In the TechRepublic Premium research mentioned above, one of the major hurdles respondents said they faced was a lack of skills and knowledge to make microservices work in their organizations—but what kinds of skills are needed?

Keep in mind that microservices aren't a single thing: They're a design philosophy, so there won't necessarily be certain programming languages or hard skills you need to know. Instead, it's all about knowing where you're at, where you want to go, and how microservices can get you there.

In an article for TechRepublic, Patrick Gray said "the best place to start simply is when acquiring new software or platforms," which is a perfect time to think of how those platforms could benefit from microservices.

Find simple, particular, use cases for microservices that have a single business objective to meet and can liberate coworkers or customers "buried under spreadsheets, macros, and Access databases trying to essentially marry information from various services in your company," Gray said.

You'll also want to have a solid grasp on DevOps, which Microsoft said goes hand-in-hand with microservices. "DevOps practices, such as Continuous Integration and Continuous Delivery, are used to drive microservice deployments," said Microsoft's Sam Guckenheimer.

"DevOps teams encapsulate individual pieces of functionality in microservices and build larger systems by composing the microservices like building blocks," Guckenheimer said. If your organization hasn't tried to deploy software using a DevOps model, you may want to start there first.

If you're already working with the communication, collaboration, and cross-team integration of DevOps, you're already halfway to understanding microservices. All that's left is to move the same philosophy of how to organize developers into the realm of organizing software.

TechRepublic®

ABOUT TECHREPUBLIC

TechRepublic is a digital publication and online community that empowers the people of business and technology. It provides analysis, tips, best practices, and case studies aimed at helping leaders make better decisions about technology.

DISCLAIMER

Cover Image: iStockphoto/Aquir