Build out your microservices approach

Nanoservices: Where they fit—and where they don't

Joydip Kanjilal
Consultant, Independent

Just as microservices represented a big improvement over monolithic architectures, nanoservices are a step up from microservices—for some things. Nanoservices don't completely replace microservices; in fact, you may have heard people say that nanoservices aren't ready for prime time and should be avoided.

Nonetheless, they're a good choice when your microservices get too big, or when you need greater isolation and flexibility than what microservices can offer. You just need to understand the pros and cons of nanoservices and where they might be a good fit in your organization. Here's how to do just that.

Not too small: The advent of nanoservices

As great as microservices are, there are downsides to using them. Nanoservices address some of those shortcomings.

While microservices start out smaller than monoliths, they can grow to be as large as the system they're meant to replace. There is no standard definition for how big a microservice should get.

For its part, a nanoservice is a small, independently deployable, testable, reusable component that essentially splits a microservice into several small pieces. Unlike a microservice, a nanoservice doesn't typically represent an entire business function.

Nanoservices contain all of the features of a microservice—they're just smaller. Think of a nanoservice as an extra-small microservice that consists of a piece of code— usually a method—that is reusable and can be exposed over the wire as a single API endpoint. And unlike microservices, nanoservices are designed to perform just one task at a time.

Nanoservices vs. microservices

Nanoservices are smaller, more isolated, and more focused than microservices. Because of that, you don't have to worry about frequent releases; they will have less impact because nanoservices have fewer things that can break. And when working with nanoservices, you can have different teams working on services concurrently.

As with microservices, there is no consensus about what size a typical nanoservice should be.

For an example of how you might use nanoservices, consider a service we'll call PayrollService, which comprises three individual services: EmployeeService, LeaveService, and DepartmentService. The EmployeeService is responsible for handling employee details and is a microservice with its own data store. The other two services could be microservices as well.

Now suppose you need a reusable method to verify employee email addresses. You can create a method that validates email addresses and wrap that method inside a service. The name of this service might be ValidateEmailAddressService, which you can expose using a REST endpoint. That's an ideal use case for a nanoservice.

Nanoservice anti-patterns to avoid

Nanoservices have several advantages. They're more granular than microservices and have a faster feedback loop, allowing you to detect and fix issues quicker. And when working with nanoservices, you can write your business logic in a function and encapsulate it as a serverless function so it can be reused.

But while the additional isolation that you can achieve using nanoservices might seem enticing, they also have their unique set of issues and challenges. If not used properly, a nanoservice can become an anti-pattern.

Nanoservices are fine-grained, with scattered, fragmented logic. They are essentially miniature, more focused versions of microservices that are independently developed, tested, released, and versioned.

Keep in mind that the more granular your services are, the more services you have to manage and the more network traffic you have. While an increased number of services is beneficial in terms of scaling and changeability, you also have to consider network complexities, maintenance, and other factors.

Also, complexity increases proportionally as the number of nanoservices in use increases.

The future of nanoservices

While they have their limitations, nanoservices hold promise as serverless computing proliferates. The fine-grained nature of nanoservices makes them good candidates for use in serverless architectures; they are small, reusable, and easily deployable.

That said, nanoservices are still in their infancy, and it is too early to predict the extent to which developers will use them in the years to come, and whether nanoservices will become the dominant paradigm for serverless computing.

Take a hybrid approach

Rather than choose between microservices and nanoservices architectures, take a hybrid approach. If your application can work with a microservices architecture, it can work with nanoservices as well.

Use nanoservices when you would like to use serverless architectures. If your application has a lot of functionality, you might want to split the functionality into several nanoservices for granularity. For all other purposes, you can use microservices.

There isn't any reason why you can’t use a combination of these architectural styles in the same application. This would help you eliminate the downsides of each while at the same time designing a high-performance, robust, and scalable architecture.

Keep learning

- **Take a deep dive into the state of quality** with TechBeacon's Guide. Plus: Download the free World Quality Report 2021-22.
- **Put performance engineering into practice** with these top 10 performance engineering techniques that work.
- **Find to tools you need** with TechBeacon's Buyer's Guide for Selecting Software Test Automation Tools.
- **Discover best practices** for reducing software defects with TechBeacon's Guide.
- **Take your testing career to the next level.** TechBeacon's Careers Topic Center provides expert advice to prepare you for your next move.