thenewstack.io

Miniservices: A Realistic Alternative to Microservices

9-11 minutes

There's no doubt that microservices is one of the hottest trends in modern software development. Everyone loves to tout they're using it but are they really? Upon further review, the lack of understanding of event-driven architecture may have more teams using the more newly coined "miniservices" to fit business needs.

Are You Really Using Microservices Anyway?

Martin Fowler's prevailing definition of microservices architecture should meet the following standards to earn the term:

- Be independently deployable and scalable executable
- Have a single responsibility
- Be loosely coupled

Ross Garrett, vice president of marketing at Cloud Elements, told The New Stack that most teams are working to decentralize as much as they can, but they aren't using as many microservices as they think.

He says that microservices architecture usually has a message-based system in which microservices could communicate asynchronously with each other, and that communication layer is unfamiliar to a lot of modern developers these days. This has seen them reverting back to using HTTP.

“HTTP as a simple request-response Web API is simplistic, it’s well understood, everyone understands how to use RESTful ways to communicate,” Garrett said, but “That flies in the face of what Martin Fowler refers to as loosely coupled” because “in HTTP, the services have to know about each other so you have to code the interfaces around each service.”

Fowler’s infamous book contends that these services should exist totally independently. An HTTP-build service has to know more about what’s going on around it in order to communicate, and it needs to know more than can be defined as a microservice.

“In a true microservices architecture, each service should have zero awareness of the services around it and, in order to achieve that, a very particular communication pattern must exist of publish-subscribe, published in a messaging queue so other people can retrieve them, the loosest coupling,” Garrett explained.

“However in app development, not a lot of developers are familiar with the pub-sub, message queue integration pattern, so they revert back to REST APIs as a way to make those services communicate. And then because they use REST, they have to know all the services around you and you lose the loose coupling,” he continued.

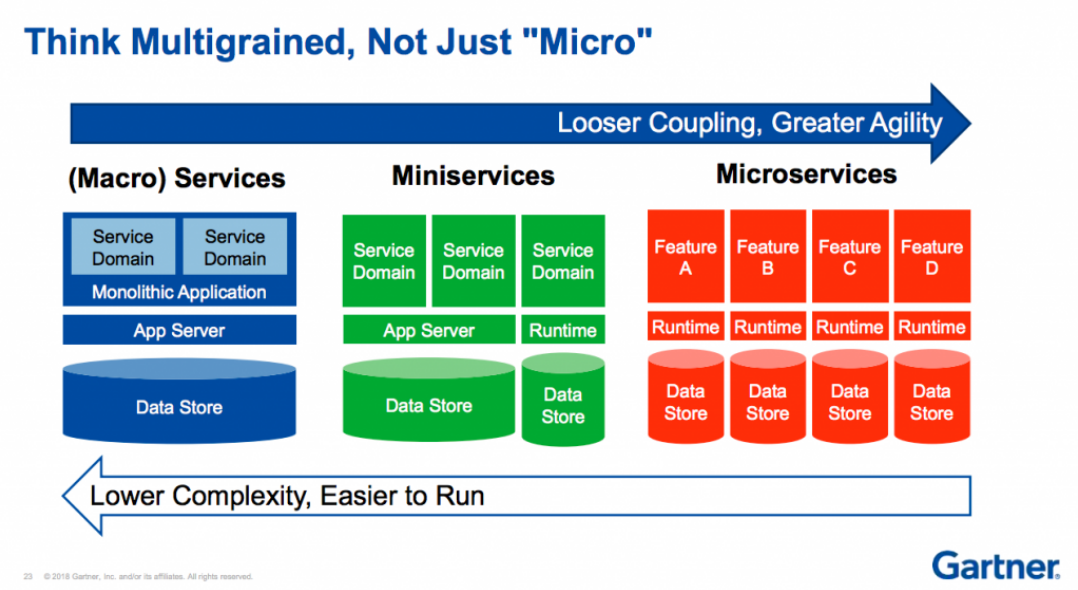
Garrett says developers then break up those apps in the smallest functioning independent components that still allow you to communicate, but they can’t be fully independent if they are using HTTP.

And Garrett says that’s OK. It’s time to embrace “miniservices,” which most companies are using a lot of anyway. He says scope or size don’t matter so much as a miniservice simply doesn’t conform to the purist’s view of microservices.

> “[A miniservice is] like a group of microservices come together in a little pattern.”

— Ross Garrett

“It’s almost like a group of microservices come together in a little pattern because they have to know about each other,” Garrett said.



Source: Gartner

Arnaud Lauret, the self-described API Handyman, drew the line between the two this way: “A miniservice seems to be a simpler, pragmatic way of doing microservices or something closely related. For example, each microservice must handle its own data, miniservices may share data.”

> “Each microservice must handle its own data, miniservices may share data.”
> — Arnaud Lauret

This all falls into the world of event-driven architecture, reacting to the events that they care about with the request-response pattern. Garrett explained these services are action like: “I have to tell you something, I can’t just receive a random event,” which then moves away from the intention that microservices architecture really looks like.

He says that if you ask an organization if they are using microservices, the answer is always “Of course.”

“It’s more about developers’ understanding of integration patterns and technologies than it is people’s desire to build a miniservice,” Garrett said. “I would say, unless you’ve moved beyond HTTP as a way for these services to communicate, you are doing miniservices.”

Use Cases Before Architecture

In the end, it doesn’t really matter if you are doing micro, mini, or even monolith, it’s about the business impact of your architectural decisions. If you can get the solution you need without having to reach the purist’s view of microservices, it shouldn’t matter.

> “Don’t confuse architectural perfection with business value.”
> — Ross Garrett

Garrett says it’s all about making the correct business decision, understanding why you are trying to make a change, and understanding the problem you are trying to solve.

Gary Olliffe, research developer at IT analyst firm Gartner, recently gave a talk entitled: “You Are Not Netflix: How and When to Use Microservices in the Enterprise.” Like its name suggests, Netflix, as one of the most architecturally forward-thinking companies in the world, has specific challenges that mostly microservices can address, but, just because it works for them, doesn’t mean it works for you. Netflix has to understand the different endpoints they need to expose their services with and the wide array of different devices with different constraints to expose them on. Often microservices is the right choice, but in many cases they are still using HTTP.

Garrett says this is “not the purist’s view of microservices, but it was functionally delivering the right goal. The decision point for me is partly then how do I solve this particular problem and what skills do I have to solve it.”

LaunchAny digital transformation consulting founder James

Higginbotham looks at this acknowledgment of microservices, which he calls “modular monoliths,” as a step in the right direction.

“It’s exciting to see modular thinking — along with loose coupling and high cohesion — to reenter our software design. Microservices is an extreme that is a fit for a few. The lessons from it are a fit for many.”

“Organizations have a choice: build modular monoliths or build with microservices. Teams can distribute the effort across both effectively, but the principles of microservices architecture help to manage the challenges that come with complex solutions,” he said. “The downside of microservices is that they push complexity to the edge. The trick is to find the right balance of where you leverage the complexity to make your organization more effective.”

When to Use the Miniservice vs. Microservice vs. Monolith

Joseph Cooper, founder and CEO of microservices tool provider KintoHub, shared some lessons from his 15 years of experience in programming backend solutions for the gaming industry, which he says has a unique set of problems that can be addressed by an architectural hodgepodge of miniservices, microservices and the monolith.

For Cooper, miniservices is all about performing one function as a service. “A miniservice is a single function as a service,” he said. He offered the example of “when you have lambda and Google functions and you want to ensure maximum cost efficiency, you don’t want to boot up an entire full-fledged feature, but simply the functionality you’re interested in.”

He says miniservices are solid solutions when you have a higher complexity, like image processing, which he says is “a very simple program to write that doesn’t require any communication between

other services."

On the other hand, "For microservices, the benefit is you have more of your code within your product. They are less cost efficient compared to miniservices because you have to have multiple functions load up in real time or you have to have them running at all time."

He continued that "Sometimes they don't come in a serverless flavor, [your services are] running the whole time. The cost efficiency is lost but the benefit is more functionality in a single project, making them much easier to work with and understand."

If you have a project with one functionality in multiple code repositories, microservices are a reasonable solution. On the other hand, Cooper said if you are running an e-commerce website or a game, where you can have ten or more functionalities within one repository, sometimes a feature has many more functionalities to complete, and a miniservice suits this need better.

And don't forget that it's not just the trending microservices and the emerging (or realizing of) miniservices that are solutions. For instance, fishing game EatMe.io would have up to a hundred global, concurrent players in a room, with games running across 12 different Amazon Web Services regions, making scaling their biggest challenge. Add to that, out of 15 games made, only seven came to life. To rapidly experiment and iterate, they would prototype on a monolith and then break down into miniservices or microservices.

"Better [to build] a proof of concept on a monolith because it's easier to manage. Then after your proof of concept and you have an idea of what your MVP [minimal valuable product] is, then you can start over and build it with microservices," Cooper explained. "Sometimes you don't know what your product is going to look like and if you were building it on microservices, you would be investing

too much time into a proof of concept.”

He also said if new versions of a game aren’t going to come out anymore but they wanted to maintain the current version, they would leverage miniservices to maximize cost saving before the game went to pasture.

Feature image by Mona Eendra on Unsplash.