Search TechTarget

5

How microservices patterns made Uber's architecture perform better

How did Uber manage to deal with the inevitable Halloween rush? Applying microservices patterns to their architecture played a big part.

George Lawton

Enterprises are turning to microservices and applying microservices patterns in order to properly develop their application architecture. The core idea is to break monolithic applications into smaller chunks of code, which in theory are easier to manage and develop independently of one another. This sounds great in theory but can spiral out of control without a coherent strategy, said Susan Fowler, site reliability engineer at Uber, at the

O'Reilly Software Architecture Conference in San Francisco. Fowler elaborated on how she went about bringing order to Uber's microservice architecture.

Uber had about 1300 microservices when Fowler began investigating how they could apply microservices patterns and improve reliability and scalability. She started a process of standardizing the microservices which allowed Uber to manage the big Halloween rush without outages. Fowler said, "We have thousands of microservices at Uber. Some are old and some are not used anymore and that became a problem as well. A lot of work has to be put into making sure you cut those out and do a lot of deprecating and decommissioning."

Microservices don't live in isolation

Microservices are parts of large and complex distributed systems that are often container hosted. The more distributed the system the more ways it can and will fail, increasing the need to use tried and true microservices patterns. Each new dependency or technology introduced adds a new point of failure. With 100 microservices this can get messy.There is a myth that microservices are like the wild West, were developers have free reign over architectural decisions, programming languages, or databases. Fowler said, "Developers think microservices allow them to code to just get the job done. Developers often hear that the goal is to build a service that does one thing well and that they can do whatever they want. This is not practical or feasible from an organizational perspective."

This can create challenges around technical sprawl and debt. Developers might adopt their favorite tools and languages, deploy with custom scripts, and adopt a custom build chain. Since there are 1000 ways to do each thing the organization ends up with 1000 ways to do one thing. Fowler said, "Any time you switch to something like a build container, you need to make sure this is done in a way that is productive and not compromising services and the availability of the overall system."

When organizations are driven by fear and quickly adopt an ad hoc approach to micro service development, developers don't know about the other microservices. They don't know if the dependencies are reliable. Sometimes there aren't even agreements on the same team. When Uber calls microservices production ready, all of the dependencies and clients can trust it because they know proper microservices patterns have been applied. Fowler said, "This is a really high goal. We will never have everything production ready, but you can get there."

The dangers of individual standards

One approach to standards is to start with local standardization. Developers figure out what requirements are appropriate for each individual service and then build from there. This was the approach taken at Uber in the beginning.

"There were a lot of problems," said Fowler. The services worked great, but they could not trust other services. When one service changed it had to change for everyone. This approach was also not scalable, since the site reliability engineers could not go to every team and determine the standards for each microservice.

Connect global standards to business metrics

A much better approach is to develop global standards that apply to all microservices, said Fowler. These need to be general enough to apply to every microservice and yet specific enough to be quantifiable and produce measurable results. These standard requirements also need to come back to key business metrics. One approach might be to stipulate service level agreements for availability between microservices as a way to measure trust. This is easy to measure but hard to code for.

A much better approach is to look at what principles lead to availability, which includes stability, reliability, scalability, performance, fault tolerance, and documentation. These are not useful on their own, so Uber came up with quantifiable requirements that relate to these. These also need to be linked to high-level business metrics like customer page views. These metrics need to be translated into requests per second on a microservice.

Documentation needs to be done from the beginning or it will lead to technical debt. Fowler found that when she talked to the team members, no one could draw the same architecture. Collectively, they would know what things look like, but it was rare that a majority would know how a particular microservice fit into the overall architecture. Without this kind of understanding it is difficult to build good microservices.

The three key steps for implementing microservice standards at Uber included group buy-in, determining organizational production ready requirements, and making production readiness part of the engineering culture. There need to be quantifiable requirements that can be tested.

"It is a long process but makes a big difference," said Fowler. "Developers all want to make the best thing they can. Standardization is not a gate, it is not a hindrance. It is something you can hand developers, saying, ‘I know you can build amazing services, here’s a system to help you build the best service possible.’ And developers see this and like it."

Prev

1

Next

George Lawton asks:

What do you think are the hardest challenges of microservices development?

Join the Discussion

Next Steps

Microservice development shouldn't be hard

Alternate JVM languages like Kotlin provide microservices developers more options

Test your microservices development IQ

This was last published in June 2017

Dig Deeper on Java DevOps

ALL NEWS GET STARTED EVALUATE MANAGE PROBLEM SOLVE