apigee

Google Cloud

Scaling Microservices

Why microservices need API management

Table of contents

Executive summary

The original vision of microservices was that they wouldn't be shared outside the small team that created them. Among the things that made them microservices, as opposed to application programming interfaces (APIs) or service oriented architecture (SOA), was the fact that developers no longer had to worry about the same level of documentation or change management that they did with a more widely shared service.

But as microservices have become popular for sharing and reusing services both inside an organization and with external partners, they've evolved well beyond the original use case of sharing functionality within a single development team. As enterprises have attempted to share microservices, many have struggled to secure them, make them reusable and adaptable beyond bespoke use cases, and understand their usage, dependencies, and performance.

Consequently, microservices have become like many other backend services that need to be managed and made consumable and reusable for developers. As is the case with these other services, this task can be accomplished by packaging and managing microservices as APIs.

Put another way, when microservices are lauded as catalysts for speed and scale, the conversation is incomplete if it does not include APIs—because without APIs, microservices cannot be securely, reliably, and adaptably scaled across an organization or to outside partners. Moreover, because these APIs and their microservices are meant to be widely consumed and reused, it's not enough for companies to merely create the APIs and move on. Rather, the APIs should be continually managed so the business can control how its microservices are accessed, generate insight into how they are used, and encourage reliability of its services.

Why microservices are popular—and challenging

Microservices. As the name implies, microservices are small and lightweight services that enable complicated applications to be broken down into components that can be developed by disparate teams. Individual components might provide payments functionality or geolocation functionality, for example.

In a microservices architecture, applications can easily leverage and reuse existing microservices components, and these components can be interconnected without fragile dependencies or tightly-coupled linkages. The potential benefits of using microservices include more developers working on new applications and digital experiences, faster development, greater agility, and more innovation.

Microservices investments have accelerated across the business spectrum—not just among big companies, a majority of which[1] are either experimenting with microservices or using them in production, but also among mid-market firms and SMBs.[2]

[1] DZone, "DZone Research: Microservices Priorities and Trends" by Anne Marie Glen, July 2018. https://dzone.com/articles/dzone-research-microservices-priorities-and-trends

[2] CRN, "Research: CIOs Up Spending On Containers, Microservices As Companies Increase Public Cloud Use" by Alec Shirkey, September 2017. https://www.crn.com/news/cloud/300092736/research-cios-up-spending-on-containers-microservices-as-companies-increase-public-cloud-use.htm?

Netflix's iterative transition from a monolith to a microservices architecture,[3] which helped the company to make its content available[4] on a dizzying variety of screen sizes and device types, is one of the most famous microservices success stories—but hardly the only one.

South American retailer Magazine Luiza has similarly leveraged microservices to accelerate the launch of new services, from in-store apps for employees to an Uber-like service to speed up deliveries. Its efforts have helped it earn praise as the "Amazon of Brazil".[5] Pharmacy giant Walgreens has likewise harnessed microservices to make its core functionality lightweight and user-friendly for developers, resulting in, among other things, more apps integrating the Walgreens customer rewards program and the distribution of billions of rewards points to customers. Other major microservices adopters, including Airbnb, Disney, Dropbox, Goldman Sachs, and Twitter, have cut development lead time significantly.[6]

But these and other noteworthy microservices accomplishments generally involve a major evolution from the original microservices vision.

Microservices were initially conceived as small, single-function services that were to be used within a single team. These microservices were meant to exist outside the kind of centralized governance and documentation characteristic of larger shared services, such as those built under service-oriented architecture (SOA) principles. However, as more businesses have found that microservices encapsulate functionality that is too valuable to be confined within a single team, they've also had to find ways to make microservices shareable.

Additionally, companies have found that the complexity of managing microservices increases with the number of microservices they deploy. Many businesses have struggled with security, with a lack of visibility into usage and performance of microservices, and with building agile microservices for clean reuse. Challenges include bolstering security in a zero-trust environment, ensuring compliance, understanding microservices usage, improving the developer experience, and encouraging and increasing microservices adoption and reuse.

---

[3] Netflix, "Engineering Trade-Offs and The Netflix API Re-Architecture" by Katarina Probst and Justin Becker, August 2016. https://medium.com/netflix-techblog/engineering-trade-offs-and-the-netflix-api-re-architecture-64f122b277dd

[4] re/-fraction, "How Netflix works: the (hugely simplified) complex stuff that happens every time you hit Play" by Mayukh Nair, October 2017. https://medium.com/refraction-tech-everything/how-netflix-works-the-hugely-simplified-complex-stuff-that-happens-every-time-you-hit-play-3a40c9be254b

[5] Bloomberg Businessweek, "Hawking TVs on Tinder Helps Fuel 2000% Rally in Brazil Stock" by Fabiola Moura and Paula Sambo, August 2017. https://www.bloomberg.com/news/articles/2017-08-14/hocking-tvs-on-tinder-helps-fuel-2000-rally-for-brazil-retailer

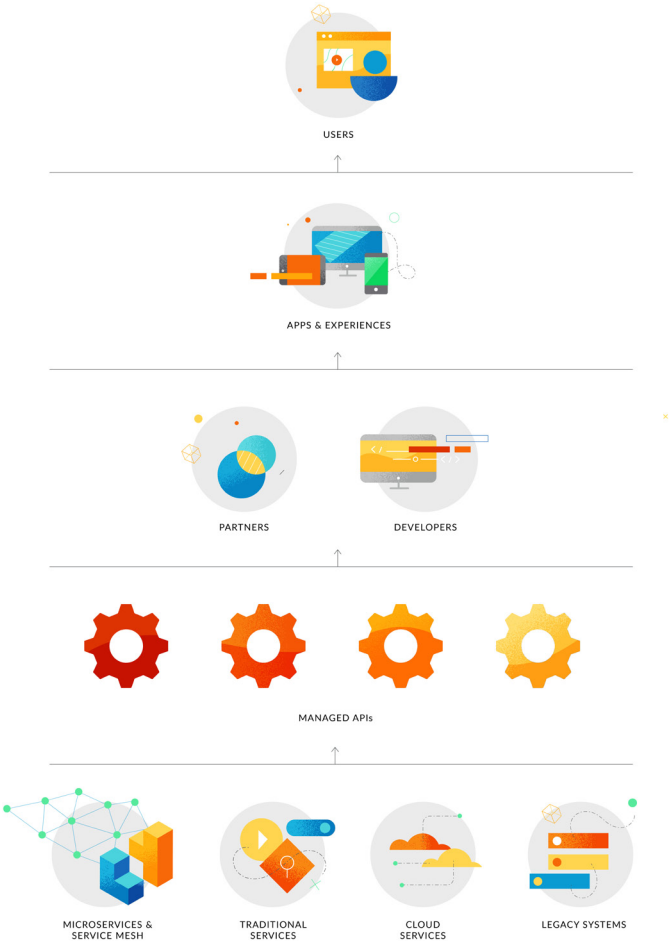
[6] Computerworld, "How to determine when and why to use microservices" by Gary Olliffe, June 2017. https://www.computerworld.com.au/article/621169/how-determine-when-why-use-microservices/

APIs make microservices shareable

Increasingly, organizations that conquer microservices management challenges and deploy microservices to business advantage embrace a common best practice: they package their microservices as application programming interfaces (APIs) to make them shareable and manageable throughout the organization and with outside developers.

"A lot of different microservices work together to provide functionality for customers," Ferry Tamtoro, CTO and VP Flex Digital Health, said in an interview. "The APIs provide a user-friendly interface for our customers so that they don't have to access every single microservice."

In another interview, T-Mobile VP of IT Chuck Knostman offered a similar appraisal: "For every microservice we build, we actually produce an API."



Microservices and APIs are not synonymous and are not interchangeable concepts. In a certain respect, microservices are similar to many other backend services that developers may need to leverage to create new applications. Because frontend developers cannot be expected to possess expertise in all backend systems or to modify their work whenever backends change, enterprises use APIs to abstract systems complexity into an intuitive and consistent interface. Microservices are no different in this regard.

APIs are the mechanism that make data and functionality, including microservices, consumable and shareable. API management platforms provide the tools to secure these APIs, optimize their reliability, encourage developer consumption with resources such as a developer portal, and adapt them for evolving use cases.

Security:
Manage microservices in a zero-trust environment

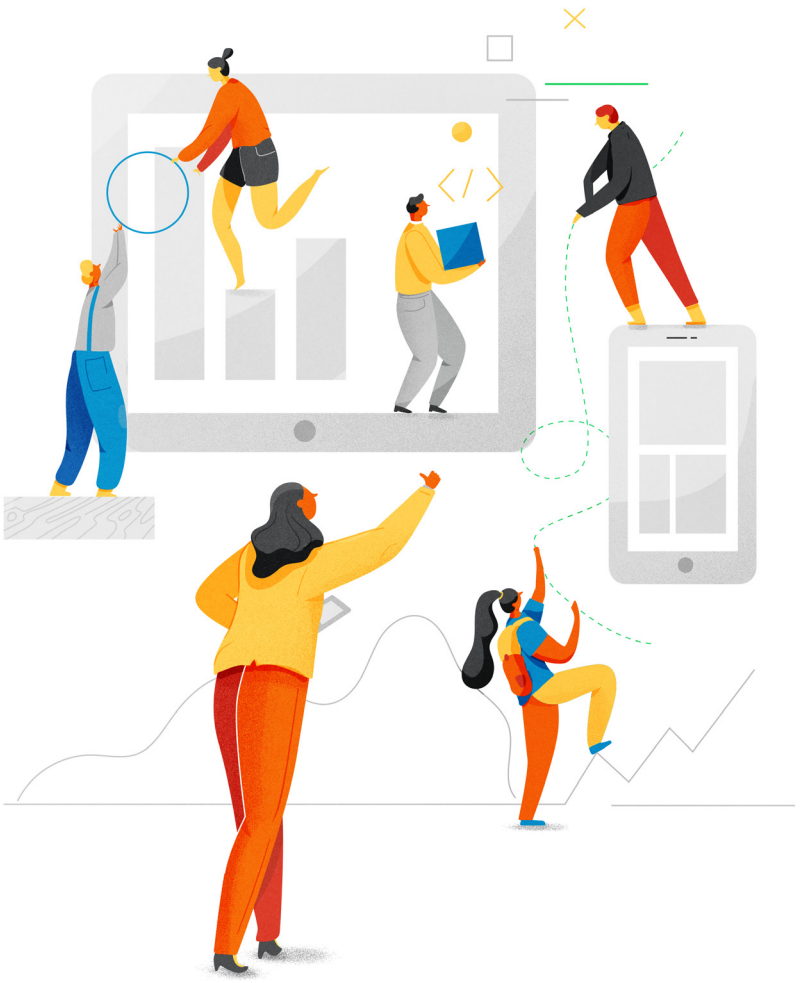
The very reasons for the success of microservices-based architectures may also present some challenges; because microservices are often developed without the centralized oversight that once existed, they offer greater development speed and agility but also potentially increase security risks. Without consistent oversight, microservices may be deployed with varying levels of security—or no security at all.

When a business packages microservices as APIs and manages them via an API management platform, the organization can implement security and governance policies, and control access to its APIs, with OAuth2, API key validation, and other threat protection features. In this way, an organization can enable microservices to be consumed and reused throughout a zero-trust enterprise environment.

Reliability:
Deliver performance and enforce SLAs

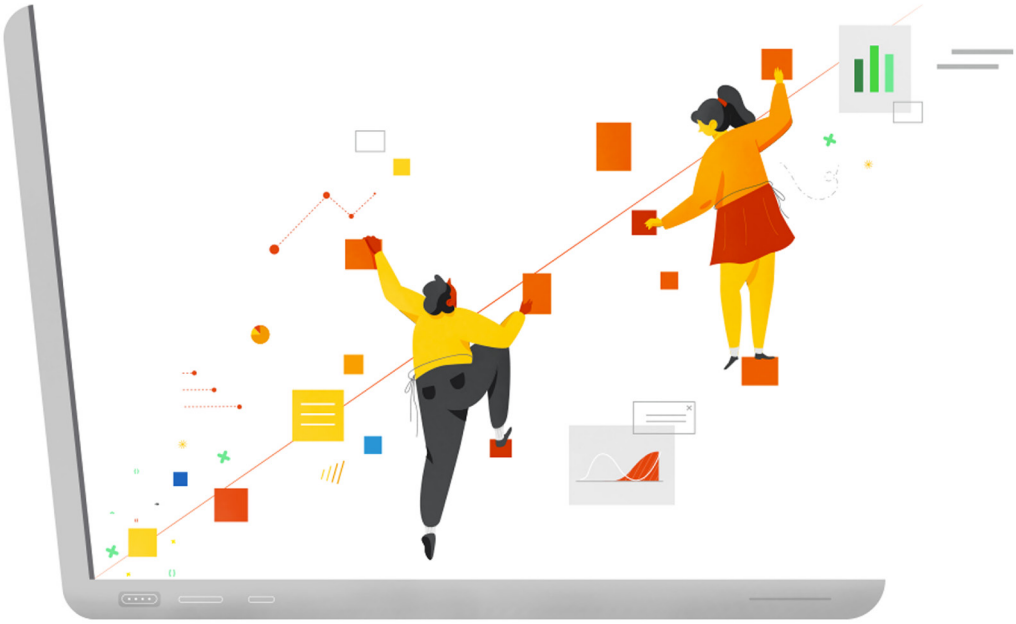
An organization's ability to deliver SLAs to consumers of its microservices may be hindered by a lack of visibility into usage and performance of the microservices APIs. This problem is especially acute when these microservices are used by disparate teams in a large enterprise or by partners and customers.

API management platforms provide the analytics and monitoring capabilities that enable enterprises to measure microservices' usage and adoption, developer and partner engagement, traffic composition, total traffic, throughput, latency, errors, and anomalies. These capabilities—along with a self-service developer portal that lets developers discover and experiment with APIs—help enterprises to give developers a first-class experience. Given that developers play a vital role in leveraging microservices for business results, their experiences may be just as important as end users'.



Adaptability:
Build agile microservices for clean reuse

Many existing and legacy services are not built for modern scale. Consequently, enterprises are adapting legacy services for modern use by breaking up monolithic applications into microservices. In most cases, however, there are already many applications taking advantage of services from the monoliths. The transition to microservices must be done in a way that makes it seamless—invisible to those other applications and developers using the monolith services.



Furthermore, as mentioned earlier, microservices are typically purpose built for a particular use case. As soon as a microservice is shared outside of the small team that created it—let alone outside the company—developers need to be able to adapt the microservices to be more widely used. When a service is meant to be shared and reused across teams and even outside of a company, it can no longer exist just as a microservice—it should be packaged as an API.

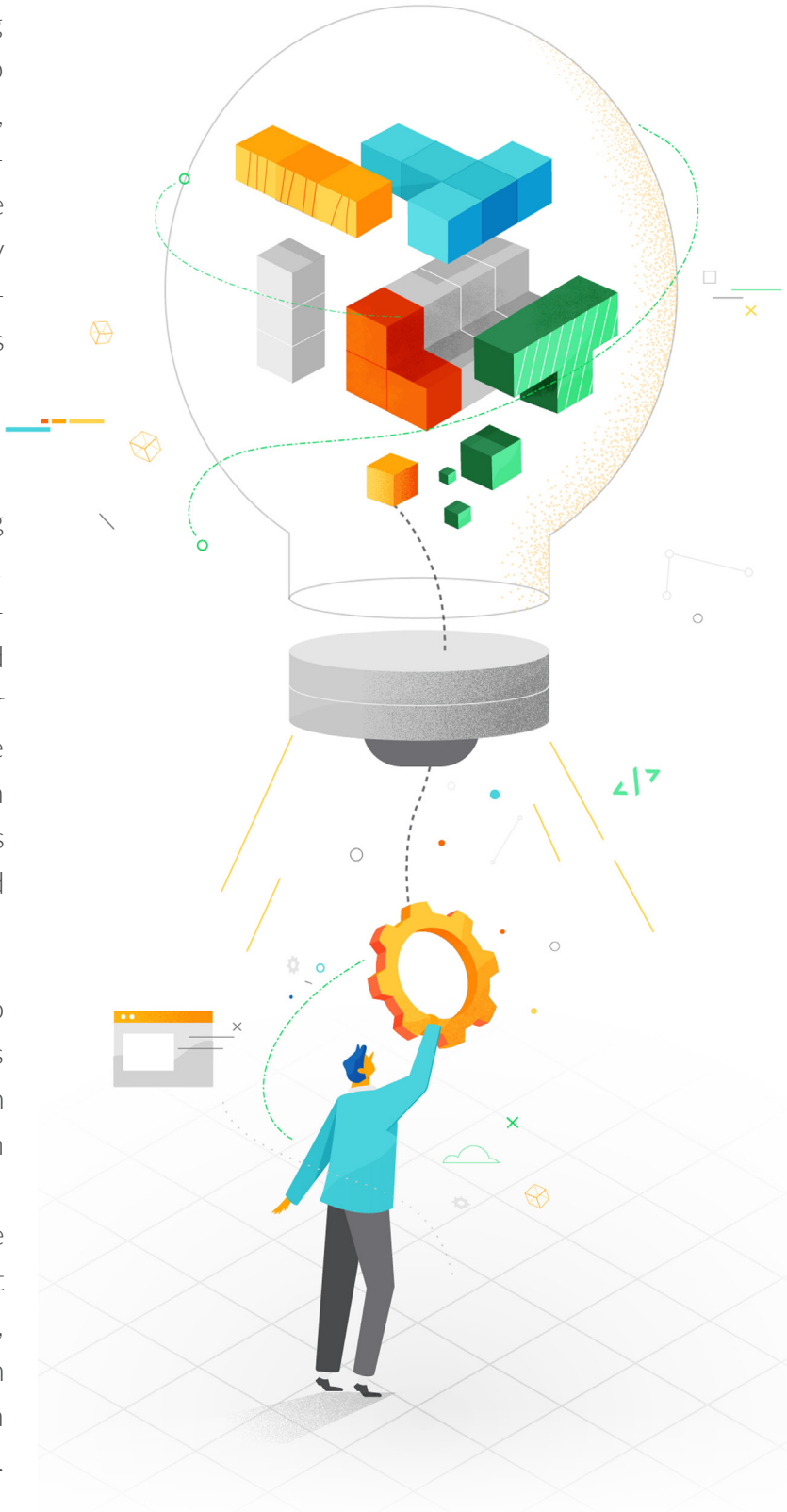
Modern APIs (RESTful, cached, and secured) can act as a facade for legacy SOAP services and monolithic applications, and securely expose new microservices for consumption. A microservice behind an API can be cleanly replaced without disruption to the applications that rely on the API. Mobile and web app developers can continue to consume an enterprise's legacy services and be unconcerned with the heterogeneous environment behind the APIs or to any transition from monolith app to microservices.

Summary

Enterprises are modernizing and scaling their applications by breaking them up into smaller components, or microservices, especially when used with cloud infrastructure. Microservices can increase software development speed and agility while enabling the reuse of shared services. The agility empowers developers but may come at a price.

When developers build microservices, they can deliver inconsistent and varying levels of security—or a complete lack thereof. Lack of monitoring and measurement can leave enterprises flying blind and unable to gauge how reliable their systems are. As microservices have become more popular and enterprises have begun to share them more widely with partners and customers, the lack of control and visibility can be dangerous.

Because APIs provide secure access to functionality and services, when a business needs to share services—whether between teams all under the same roof or with developers and partners outside the enterprise—it needs APIs. Companies are increasingly looking to API management platforms to provide the security, reliability, visibility, and adaptability of the APIs in its microservices architecture, whether in the cloud or across hybrid environments.



Now that you've finished reading,
why stop learning?

TRUTH ABOUT MICROSERVICES

Take a deeper dive into the concepts and strategies you've learned to scale microservices by exploring more success stories, videos, eBooks, articles, and more.

KEEP LEARNING

FREE TRIAL

Explore Apigee Edge, a full lifecycle API management platform that helps you manage the entire API lifecycle from design through iteration and helps you control the complexity of microservices.

TRY IT FREE

MAXIMIZING MICROSERVICES

Learn how a service mesh and API management platform can allow your team to reduce complexity and increase consumption to maximize the value you get from your microservices in this eBook.

GET THE EBOOK

apigee

Share this eBook

on social

with a colleague

Google Cloud