Microservices Done Right

Microservices need API management

#DigitalKnowHow
by apigee

Table of Contents

Executive Summary

What's a microservice?

Is it a way of breaking an application up into component services that can be scaled independently (depending on whether they need more compute resources, memory, or IO) and then having them talk to each other via API service interfaces?

Is it a great lightweight way to share and reuse services across the enterprise and with developers outside the enterprise?

The answer is "yes" on the first question—and "not exactly" on the second. The original vision of microservices was that they wouldn't be shared outside the team you scrummed with. Among the things that made them "microservices," as opposed to APIs or service oriented architecture, was the fact that you no longer had to worry about the same level of documentation or change management that you did with a widely shared service.

As microservices have become popular for sharing and reusing services widely across the organization and outside—well beyond the original use case of sharing with a single development team—companies have struggled with securing them, with understanding usage, dependencies, and performance, and with making them reusable and adaptable beyond bespoke use cases.

All microservices have APIs and enterprises need to manage their microservices in the same way as they manage APIs. API management platforms enable organizations to implement security and governance policies across all of their microservices APIs, deliver the analytics and reporting capabilities that provide deep visibility, and implement an API facade that delivers modern RESTful APIs for legacy SOAP services and exposes the new microservices securely.

Introduction

Microservices represent a new form of API-based application architecture that enables enterprises to leverage continuous iteration/continuous deployment (CI/CD) tools and processes within containerized architectures and PaaS. Nearly 70% of organizations are either using or investigating microservices, and nearly one-third are using them in production[1].

These are powerful techniques popularized by many companies, including Amazon, Netflix, and AirBnB. Amazon championed the approach and proved it to be useful in ensuring effective communication within teams and enabling the company to deploy code to production hundreds of times of day[2]. At Netflix, one of the earliest adopters of microservices, roughly 30 independent teams have delivered over 500 microservices[3].

However, companies have struggled with security, with a lack of visibility into usage and performance of the microservices APIs, and with building agile microservices for clean reuse. Their challenges include bolstering security in a zero-trust environment, ensuring compliance, providing transparent control to mitigate risk, improving the developer experience, and encouraging and increasing microservices reuse.

Managed microservices are the solution; companies are making strategic investments in API management platforms for their microservices success.

All microservices have APIs and enterprises need to manage their microservices in the same way as they manage APIs.

1. http://www.nginx.com/resources/library/app-dev-survey/
2. http://www.thoughtworks.com/insights/blog/case-continuous-delivery
3. http://blog.smartbear.com/microservices/why-you-cant-talk-about-microservices-without-mentioning-netflix/

Security: Manage Microservices in a Zero-Trust Environment

Microservices. The name implies small and lightweight services. Microservices enable complicated applications to be broken down into components that can be developed by disparate teams. Applications can easily leverage and reuse existing components, and these components can be interconnected without fragile, complex dependencies or tightly coupled linkages.

The very reasons for the success of this architectural model also present some challenges. Developers are building APIs and microservices without the kind of centralized oversight that once existed. They are deploying them more widely than ever before. They implement inconsistent and varying levels of security—or no security at all.

When developers deploy microservices in the public cloud and neglect to deploy common API security standards or consistent global policies, they expose the enterprise to potential security breaches.

Companies must assume a zero-trust environment. An API platform enables enterprises to implement security and governance policies like OAuth2 across all of their microservices APIs.

TrustPilot uses the Apigee API platform to secure its microservices, and doesn't have to distinguish between internal and external use. The company also takes advantage of the platform's authentication, caching, and analytics capabilities. TrustPilot built its microservices in the AWS cloud, and all microservices communicate via REST APIs. The company's developers access the microservices, as do external partners and customers.

TrustPilot

TrustPilot is a leading European review site, where consumers rate their customer experiences with merchants online. The company wanted to boost revenues from its merchant customers by offering a variety of new products based on the consumer reviews data. When a merchant customer accesses TrustPilot data through an API, it calls as many as five different microservices.

TrustPilot started its foray into APIs with the simple goal of exposing its data to customers and partners. But its perspective shifted when the advantages of a microservices architecture became clear.

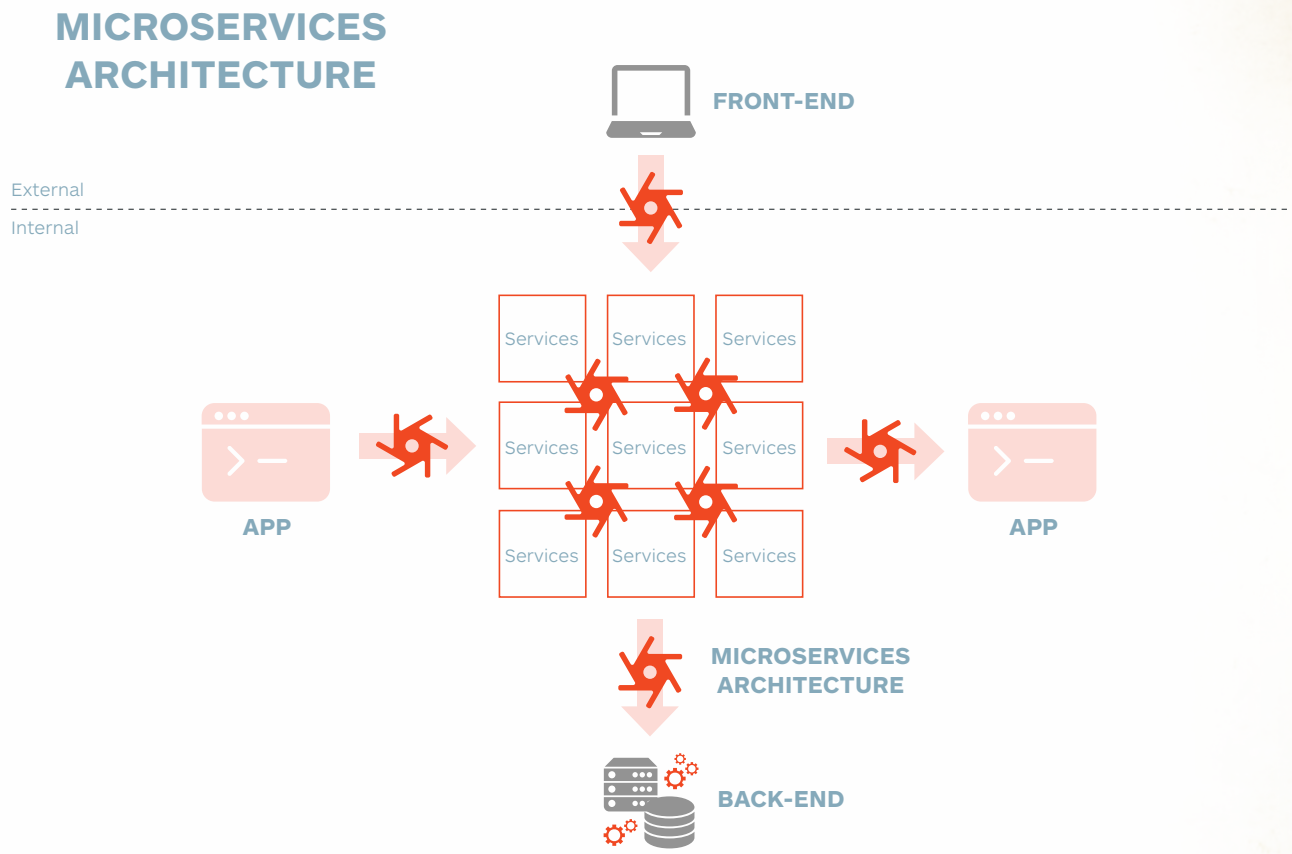
“The primary objective became to build services and products that scale and are decoupled from each other,” said Ole Dallerup, the company’s vice president of engineering. “Microservices architecture enables us to move extremely quickly, and deliver products much quicker.”

The company’s main focus was enabling easy and deep integration for partners and customers. It needed to secure its microservices, as it didn’t want to distinguish between internal and external use.

TrustPilot also wanted to gain better visibility into product usage and data, Dallerup said. The company needed a platform that could deliver “analytics and tracking at high scale out of the box.”

Reliability: Deliver Performance and Enforce SLAs

Developers have found that the same lightweight API services that have proven to be resilient, scalable, and agile for front-end, back-end, and application-to-application scenarios can also be leveraged for application assembly. This is another reason for the rise of microservices architecture.

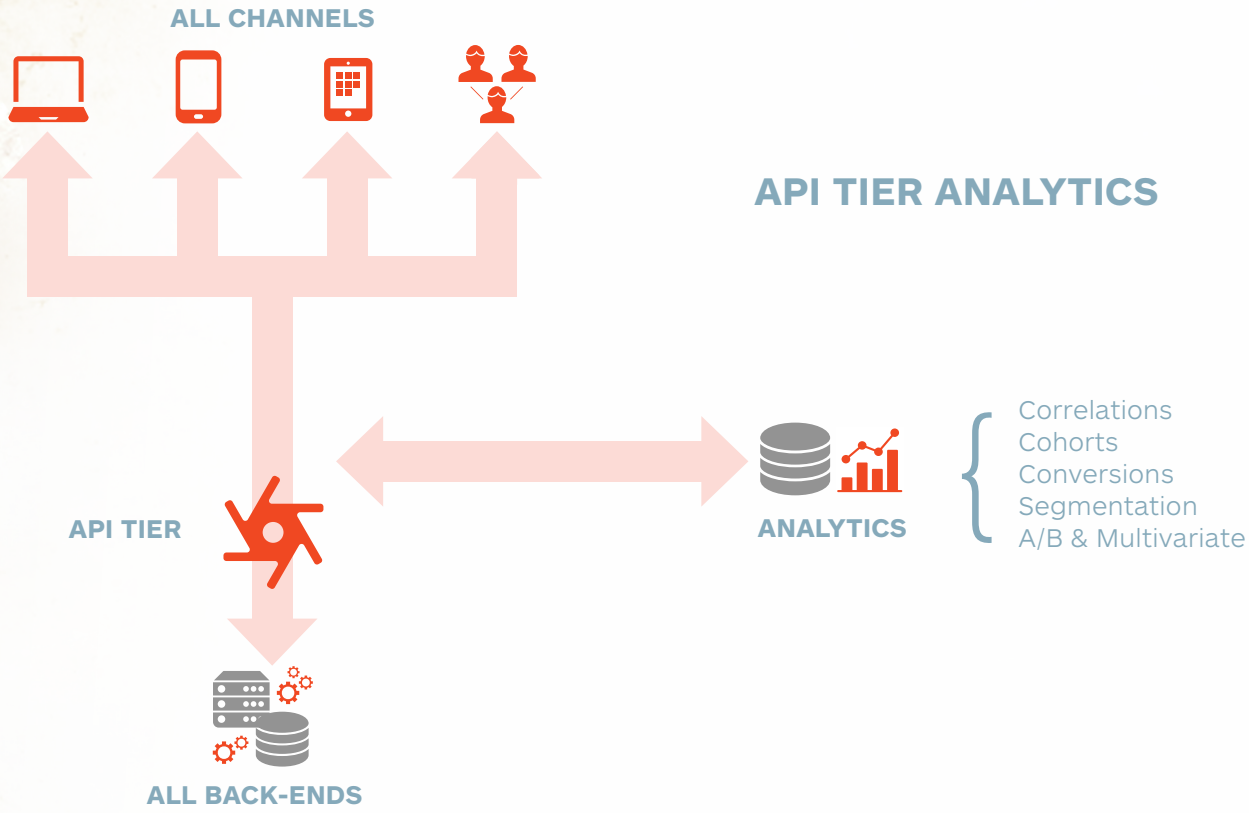


By definition then, microservices means building dependencies between your software. All of your microservices depend on all the rest of them. This raises inter-dependency problems not unlike those that exist for SOA.

Which service is talking to which other service? Which service is dependent on which? Which services are performing, and which are not? These are important questions to answer in order to understand the reliability of your infrastructure, where reliability is dependent on usage, a complex matrix of dependencies, and the services' performance.

An organization's ability to deliver SLAs to consumers of its microservices is 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 reporting capabilities that enable enterprises to measure microservices' usage and adoption, developer and partner engagement, traffic composition, total traffic, throughput, latency, errors, and anomalies.



Magazine Luiza, a Brazilian retail firm, is an example of a company for which microservices were the best way to rebuild its e-commerce system for the cloud. The company built a set of APIs so that mobile apps and partners could be easily connected to their e-commerce services.

Magazine Luiza uses the Apigee API platform to host API proxies for its mobile apps and web apps, to get fine-grained visibility into the consumption and performance of the services, and to enable a consistent security framework for all of the retailer's microservices and legacy services.

API management platforms provide analytics and reporting to measure microservices

- Usage and adoption
- Developer and partner engagement
- Traffic composition
- Total traffic
- Throughput
- Latency
- Errors
- Anomalies

Magazine Luiza

Magazine Luiza is a fifty-year-old retailer in Brazil with more than 700 stores, eight fulfillment centers, more than 20,000 employees, and more than 43 million customers. The company needed to quickly deliver a range of new capabilities, including one-click buy, buy online, pickup in the store (BOPUS), personalization, and partner enablement. To do so, it replatformed its e-commerce system with microservices and cloud.

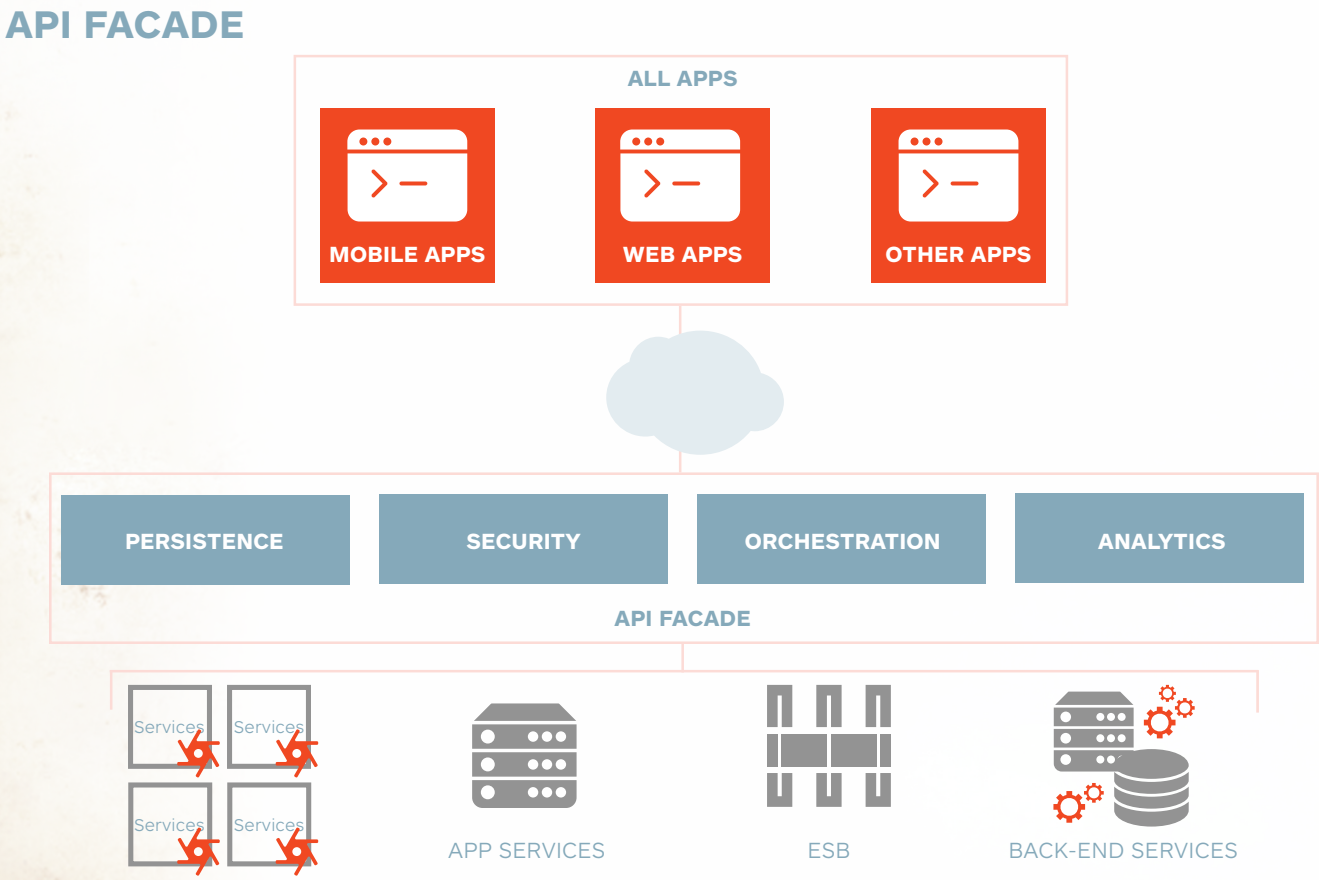
Magazine Luiza had a set of legacy APIs for the web and its mobile app, but these APIs were accessing a monolithic app built with 150,000 lines of code. The deployment of new APIs was slow and cumbersome: there were many dependencies, they were difficult to scale, and responsibilities were distributed across many teams. The company adopted an API management solution to get better visibility into usage of its services.

Adaptability: Build Agile Microservices for Clean Reuse

Many existing and legacy services are not built for modern scale. Consequently, enterprises are adapting legacy services to be used as microservices, and are decomposing monolithic applications. In most cases, however, there are already many applications taking advantage of services from the monoliths. So 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, microservices are typically purpose built for a particular use case. But as soon as your microservice is shared outside of your "two pizza team"—or even outside the company—developers need to be able to adapt the microservices to be used more widely. What's a service that's meant to be shared and reused across teams and even outside of your company? It's an API.

An API platform acts as an API facade—delivering modern APIs (RESTful, cached, and secured) for the legacy SOAP services of the monolith apps and exposing the new microservices securely. Mobile and web app developers can continue to consume an enterprise's services and be oblivious to the heterogeneous environment or to any transition from monolith app to microservices.



Belly, a mobile loyalty and rewards platform, adopted an API management platform to build an API facade that sits in front of its backend legacy services and microservices. The API facade provides caching, traffic management, security, visibility, and orchestration of services in the backend, all in a way that's invisible to the mobile app developer who is building on Belly's platform. The company can also build custom API interfaces for each mobile consumer, whether Android, iOS, iPad, or other external partners.

Belly

Belly is a fast-growing mobile loyalty and rewards platform. To speed innovation, the company adopted microservices for developing new capabilities for its mobile apps.

However, its mobile app still had to access several legacy services, including user authentication. The company also had a single Ruby on Rails app that acted as an orchestration layer for its backend legacy services and new microservices. The Rails app also provided a layer of security that enabled access control, but it was slow and it didn't scale well.

Belly now deploys the Apigee API platform and uses an API facade in front of its backend legacy services and microservices. An API proxy running in the API platform calls both the legacy services and new microservices as target endpoints.

Summary

Enterprises are scaling up their applications by breaking them up into smaller pieces, especially when used with cloud infrastructure. Using microservices, companies enjoy the benefits of agility and speed when building software, while enabling the reuse of shared services. These smaller services result in simplicity and complexity at the same time. The agility empowers developers, but this can come at a price.

Developers 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 become popular, they are being shared more widely inside of enterprises and with more and more partners and customers. When you're talking about sharing services—whether between teams all under the same roof, or with developers and partners outside the enterprise—you're talking APIs. All microservices have APIs and companies are increasingly looking to API management platforms to provide the security, reliability, visibility, and adaptability of the APIs in the microservices architecture.

A distributed API management platform uses federated microgateways to co-locate APIs with their microservices and applications environments, while keeping the other (non-gateway) API management services centrally located. The distributed API runtime enforces consistent security and governance policies across all microservices and collects API data for analytics. This deployment model provides enterprises a single pane of glass to manage all microservices APIs across microservices stacks and clouds.

About Apigee

Apigee® powers the APIs that make every business a digital business. Apigee provides a leading API platform that helps companies—from disruptive start-ups to the Fortune 100—rapidly adapt to the business and technology requirements of the connected, digital world. Many of the world's largest organizations select Apigee to enable their digital business, including over 30 percent of the Fortune 100, four of the top five Global 2000 retail companies, and five of the top 10 global telecommunications companies.

For more information, visit apigee.com

Share this eBook