BROADCOM®
SOFTWARE

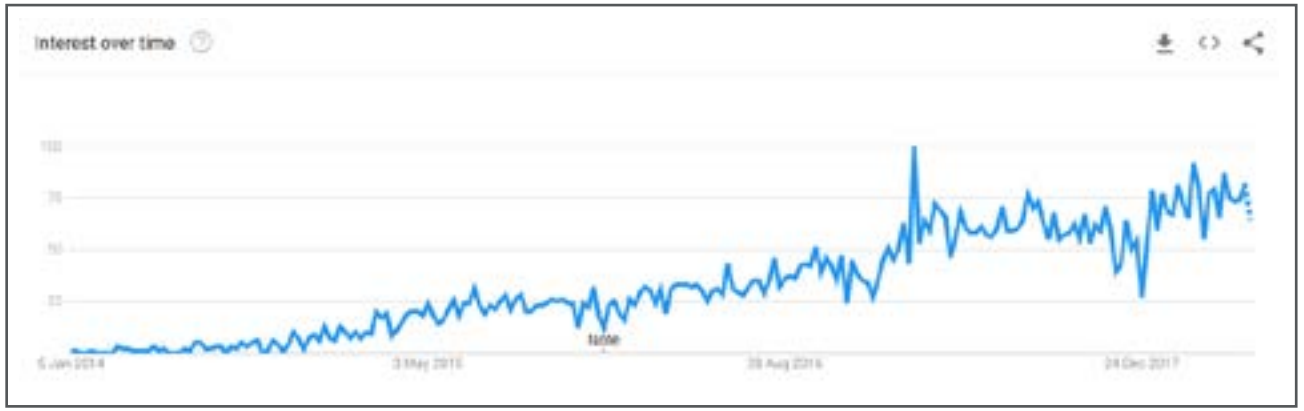
Microsurfaces: The Role of APIs in a Microservice Architecture

Microservices Adoption Continues to Grow in All Industries

According to a recent survey, 86 percent of respondents expect microservice architecture to be their default approach to building software systems within five years.

There are many reasons for moving to a microservice architecture, from increasing delivery speed to improving evolvability and scalability. However, there are also potential pitfalls as complexity shifts from place to place in the system and the associated organizational landscape. As a result, organizations moving to a microservice architecture need tools and techniques to help them deal with these new challenges.

When it comes to microservice technologies, people have called containers the **"gateway drug to microservices,"** and containers can certainly ease deployment and operability. However, API-based communication may be even more vital to delivering value in all aspects of a microservice architecture. This ebook explores the varied roles that APIs play in the microservices landscape.



APIs vs. Microservices

Before analyzing the value of APIs in a microservice architecture, it helps to clear up any confusion between the definitions of APIs and microservices. When someone says they are "building an API," they are likely developing a microservice with an API front door. Conversely, there are some who refer to API facades for legacy services as microservices. Both examples blur the lines between the terms.

What is an API?

- An application interface exposedon a network
- Can use many styles (e.g., REST), protocols (e.g., HTTP) and data formats (e.g., JSON)
- Can be implemented in many ways (standalone microservice, monolithic application, API gateway, ESB)
- The consumer's view of a service

What is a microservice?

- An independently deployable application component
- Often self-contained (e.g., interface, logic, data)
- Exposes functionality to other services and external clients via APIs
- The provider's view of a service

It is important to remember that the API is just the interface, and even more important to remember that the service consumer views the API as a contract, and they don't care what's behind it as long as they get what they need.

Microservices Origins

In their seminal 2014 blog post, Microservices: a definition of this new architectural term, James Lewis and Martin Fowler called out API-based communication as a fundamental characteristic of microservices:

"The two protocols used most commonly are HTTP request-response with resource APIs and lightweight messaging."

Although the microservices trend spiked in popularity following that blog post, it became clear that companies such as Netflix and Amazon had been using this architectural approach for a **number of years.** Notably, these same organizations that were early adopters of microservices were also key pioneers in the **API economy.**

This is no coincidence. APIs and microservices share three common origins:

- The architectural principles of the World Wide Web
- Technological affinity with mobile and cloud computing
- A decentralized approach to process, organization and culture aligned with the agile movement

Microservices Origins Continued

The Web and Service-Oriented Architecture

The rise of Web-based computing in the 1990s ushered in new possibilities and complexities for distributed systems. Service-oriented architecture (SOA) became a popular approach to building systems, but the **SOA movement faltered** when its implementation approach became overly centralized, thus abandoning the principles of the Web. In parallel, the Web API movement offered more intuitive and task-oriented interfaces to useful Web services. Companies like Amazon and Netflix synthesized SOA's service concept with "of the Web" interfaces to form a high scale, high velocity architecture.

Mobile and Cloud Computing

A decade after the Web explosion came the rise of mobile as a desirable computing platform. This drove the need for mobile-friendly interfaces (Web APIs) to modularized services. In parallel, cloud computing gained prominence. The emphasis on automation in the cloud produced a new use for APIs, as well as removing barriers to application deployment. Cloud computing provided a novel platform for deploying more granular API-fronted application components.

Decentralization and the Agile Movement

As the capability and scale of distributed systems have increased, there has been a trend toward decentralization in both the system itself as well as the supporting organization. This is another trend that follows the path of the World Wide Web. The agile software movement arose as a reaction to the same centralized approach to enterprise IT that hampered the SOA movement. Agile's popularity and success in software development led to the CI/CD approach to software deployment, followed by the cultural philosophy of the DevOps movement. This **"agile progression"** aligned well with systems of independently deployable services communicating through Web APIs.

Once again, it is clear that APIs are a fundamental part of succeeding with a microservice architecture.

The Business Value of Microservices

There are many reasons for an organization to move to a microservice architecture, but in general they are hoping to achieve "speed and safety at scale" for their software delivery.

There are potential SDLC benefits, such as reducing the need for cross-team coordination and opening up more language options; operational benefits, like more flexible deployment and manageability; and even organizational benefits such as alignment between technology assets and cross-functional teams.

Interestingly but not surprisingly, many of the benefits attributed to a microservice architecture stem from the API-first nature of microservices, namely:

Composability

When services are published through an API, it is easier to use them in multiple business contexts to assist in various business processes

Testability

When services are accessible over a network boundary, it is easier to isolate tests and exercise individual components of the system

Evolvability

When services are exposed through an API, implementation details can be hidden from the consumer, making it easier to change components without impacting dependent parts of the system

Comprehensibility

When a complex system is broken down into modular APIs, it is easier to understand the overall business functionality of the system, which helps in both designing and maintaining the system

Automatability

Along with the data plane API benefits above, control plane APIs allow automation in the deployment and management of microservices, thus increasing the velocity of software delivery

Big Challenge, Bigger Benefit

Complexity is a given for distributed systems, and the key to dealing with it is arriving at optimal abstractions. Microservices provide the opportunity to modularize a big system effectively through service boundary definitions; however, arriving at the right boundaries is tricky.

Applying Design Thinking

API design is a mature field with proven practices. Notably, APIs designed from the consumer perspective with a focus on **jobs to be done** have greater usability and higher rates of adoption. This same approach can be used when designing a system of microservices, whether starting from scratch or breaking down an existing monolith. API design thinking helps to identify the right service boundaries and helps to establish loose coupling between services so that implementation details don't leak through.

Furthermore, designing and defining a microservices system through APIs helps make it more comprehensible for those tasked with managing the system, for those just getting started on developing the system, and other external parties like business stakeholders.

APIs are a living part of the system that can be used to intuit the functions the system and its components provide.

APIs for Managing Microservices

When organizations switch to microservices from more monolithic application architectures, their gains in delivery speed and scalability have the potential to be offset by operational complexity in managing, monitoring and securing the new distributed architecture. Monolithic app servers may have contributed to the issues that microservice architecture purports to solve, but they did provide consistency in managing these system concerns. For microservice implementers, one option in dealing with these concerns is to adopt a single deployment platform, but that uniformity goes against many of the principles and practices that have made microservice architecture effective.

A better option is to look at the APIs in the system that expose core business functionality and allow service-to-service communication as a normalizing mechanism. A system of microservices may be distributed across a variety of platforms, but by using APIs as the control point in the architecture, the platform is less significant.

API management technology is an established capability used by an abundance of organizations to provide:

- **Access control**
- **Monitoring**
- **Service level management and other system capabilities**

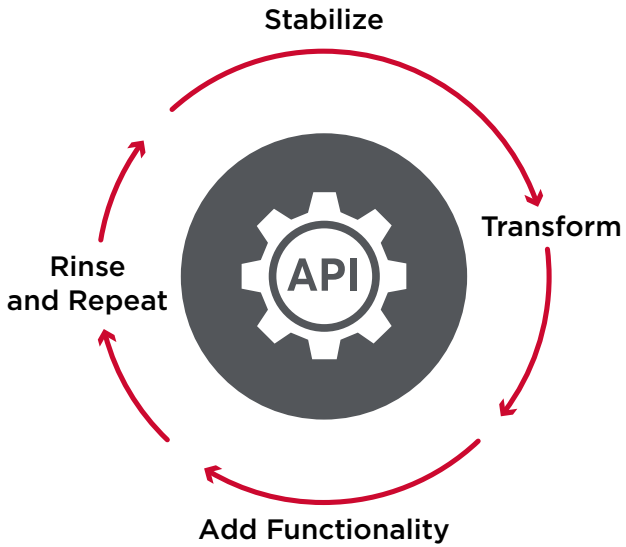
These capabilities are vital for any distributed system and are particularly important for multi-platform microservices.

APIs for Migrating Microservices

Most organizations moving to microservices have legacy systems already in place that provide the key functions of their business. When moving to a microservice architecture, these organizations need to decide the order in which they migrate these functions. In some cases—such as infrequently changing legacy systems or vendor-provided software solutions—it would not make sense or even be possible to migrate at all.

Regardless of the underlying implementation of the service—microservice, monolith, off-the-shelf software—it can be exposed through an API. For services being migrated to a microservice architecture, providing a Web API provides a loose coupling mechanism so that the service implementation can be changed without impacting consumers. For services that won't be migrated to microservices, APIs provide a means for participating in a distributed system with microservices.

Mike Amundsen's STAR method (stabilize, transform, add functionality, rinse and repeat) is a detailed approach that illustrated how APIs can be used to aid in microservice migration.



STAR METHOD ILLUSTRATING HOW APIS AID MICROSERVICE MIGRATION

Shared Evolution of APIs and Microservices

It is clear that Web APIs play a number of key roles in a successful microservice architecture. As the widespread usage of APIs becomes intertwined with the maturation of microservices, the two movements are evolving together.

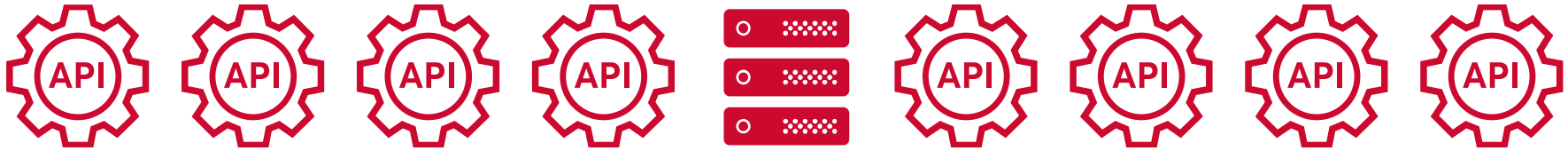
Reactive microservices systems are gaining popularity, as organizations seek to **maximize decoupling of domains**. This approach to microservices often utilizes asynchronous communication through protocols like Apache Kafka or AMQP, to avoid the synchronous nature of HTTP.

Service Mesh

The "service mesh" concept has been spawned from within the microservices community to enforce system-wide policies on microservice-to-microservice communication. Service mesh solutions may favor new protocols like gRPC and data formats like protobuf over HTTP-based =APIs based on a desire for more optimized runtime performance. Service meshes also rely heavily on **runtime service discovery**, a vital function in the ephemeral microservices landscape.

Although Web APIs may not be used explicitly in either reactive systems or a service mesh, the API industry has lessons to share in both cases. Swagger (now the OpenAPI specification) provides a universal metadata description language for Web APIs, helping with interoperability, design time understanding and automated activities like testing and monitoring. There is a similar need for asynchronous protocols, and in fact Fran Méndez has already evolved the OpenAPI concept into the **AsyncAPI** specification to address this.

The value proposition of a service mesh is similar in functionality to what an API gateway provides in an API implementation, but different in non-functional characteristics. One can conclude that other API management capabilities may be similarly valuable to a microservice architecture, most notably the design time service discovery provided through API developer portals or catalogs of services.



APIs and Microservices

APIs and microservices have a linked history and a complementary set of potential benefits. Using APIs in your microservice architecture can help incent good design, comprehensive management and ease migration of existing applications.

To dig a little deeper into these concepts, we suggest reading the following O'Reilly books for free, compliments of Broadcom and Layer7:

Microservice Architecture: Aligning Principles, Practice and Culture

Securing Microservice APIs: Sustainable and Scalable Access Control

DISCOVER HOW THE ANALYST-ACCLAIMED API MANAGEMENT AND MICROSERVICES PORTFOLIO FROM CA TECHNOLOGIES CAN HELP YOU MODERNIZE YOUR APPLICATION ARCHITECTURE WITH APIS AND MICROSERVICES AT **THE LAYER7 PRODUCT PAGE AT BROADCOM.COM**

BROADCOM® SOFTWARE

For product information please visit our website at: broadcom.com

BC-XXXXEN August 2021