IDC ANALYZE THE FUTURE

IDC TECHNOLOGY SPOTLIGHT

Brought to you by Tech Mahindra, Powered by IDC

This IDC Technology Spotlight examines the importance of microservices and open source technologies to contemporary digital transformation initiatives. It also discusses the role played by systems integrators in helping organizations harness the power of microservices and open source technologies.

Understanding the Nexus Between Open Source Technologies and Microservices

June 2019

Written by: Arnal Dayaratna, Research Director, Software Development, and Larry Carvalho, Research Director, Platform as a Service

Introduction

Today's business environment requires organizations to intensify the cadence and scope of their digital transformation initiatives to remain competitive. Confronted with a business landscape in which competitors leverage digital solutions to solve business problems, organizations from all industry verticals are transforming themselves into software companies. By enhancing their capabilities to develop digital solutions, organizations seek to demonstrate enhanced agility, velocity, and intelligence regarding their ability to respond to customer needs and evolving market dynamics. This embrace of software and digital solutions involves a recognition of the changing nature of software and, in particular, the increased importance of microservices and open source technologies.

AT A GLANCE

WHAT'S IMPORTANT

Developer adoption of open source technologies and microservices will continue to increase from 2019 to 2024.

KEY TAKEAWAY

Open source technologies are fundamental components of microservices-based applications at a multitude of layers of the technology stack.

Microservices empower organizations to develop applications as a suite of loosely coupled services that enable organizations to make rapid updates to each microservice. Open source technologies leverage a commitment to distributed peer review, transparency, and collaboration that empowers a decentralized network of individuals and organizations to contribute to an application. The decentralization specific to open source technologies creates releases, versions, forks, and product road maps that facilitate innovation. Just as microservices architectures empower distributed teams to rapidly propagate innovation to an application, open source technologies similarly empower distributed teams to innovate on existing applications and digital services.

Drivers for Use of Microservices

One of the dominant trends in contemporary application development involves the transformation of legacy applications into microservices-based applications. Saddled with large and complex legacy applications that often feature dependencies on runtime environments, obsolete languages, and monolithic architectures, organizations have taken up the challenge of transforming monolithic applications into microservices as part of broader digital transformation initiatives. Increased interest in microservices as the foundational architecture for contemporary development is a response to concerns about the following:

- The speed with which development teams can update a monolithic application
- The need to update an application with zero downtime
- The ability to decompose a domain, resulting in accelerated business agility
- The longevity and sustainability of the technologies that constitute a monolithic application
- The need to enhance and update applications with increasing frequency
- The ability of microservices to accelerate development and deliver enhanced development agility, granular scaling, and optimized resource consumption

Microservices represent the foundation of modern application development because of their ability to decompose applications into modular code components that correspondingly enhance the operational agility and velocity of development teams. Importantly, microservices empower developers to rapidly propagate changes to individual modules within an application, thereby enabling the expedited delivery of application changes and updates.

Drivers for Use of Open Source Technologies

Drivers for the use of open source technologies include the following:

- Costs of proprietary software
- The opacity of processes specific to the development of proprietary software
- Security, performance, and reliability concerns across a multitude of devices, platforms, and infrastructures
- The business continuity of software development projects given a business environment in which the original creators of a proprietary product may choose not to continue developing that product in the future

Open Source Technologies

Overview

Open source technologies are defined as technologies whose constituent technologies — whether lines of code or otherwise — can be viewed and modified without restricting other parties from distributing those same technologies. An open source software application, for example, is one whose source code can be reviewed, changed, and distributed. Open source software is widely used because it provides a pre-assembled solution that a developer would otherwise need to build from scratch. Moreover, because open source software gives developers access to an application's source code,

developers can understand how an application was conceived and architected before deciding whether it makes sense to incorporate it into their own work. This ability to control and modify an open source application represents a notable driver for the usage of open source technologies because developers acquire confidence in solutions that they understand.

Security constitutes another driver for the use of open source technologies; given that these technologies are publicly available, users have the opportunity to proactively identify security defects before they result in markedly negative repercussions. Open source projects tend to have vibrant communities that support them and that are likely to spot security-related defects. Similarly, open source developers are passionate about their reputations and have a vested interest in ensuring that their software contributions are free from defects.

Another driver for the use of open source software concerns the belief that open source projects are more likely to endure than proprietary software projects because they are freely accessible to large communities of developers, users, and contributors. In other words, open source projects are more stable than their proprietary counterparts because, theoretically, any developer or community can lead the development of an open source project in the event that the original developer stops doing so.

Further, open source is built around a culture of transparency, collaboration, and meritocracy in ways that are designed to promote healthy dialogue between members of the relevant community. Many developers and organizations gravitate toward open source technologies out of the belief that democratic dialogue and disagreement between community members lead to the improvement of the software. As is the case with science and scientists, open source developers build on the contributions of their peers and are subsequently invested in ensuring the maintenance of transparent, collaborative ecosystems for open source software development.

Illustrative Examples

Illustrative examples of open source software that can claim worldwide enterprisewide adoption include the following:

- Linux, the Unix-like operating system
- Linux Foundation, Open Network Automation Platform (ONAP)
- Acumos, an open source artificial intelligence/machine learning platform
- GNU Compiler Collection (GCC), which delivers tools and utilities that complement the Linux kernel
- MySQL, one of the world's most popular open source databases
- Apache HTTP Server Project, an open source HTTP server for modern operating systems including Unix and Windows
- The vast majority of development languages and frameworks, including PHP, Python, JavaScript, Node.js, Java, Django, Ruby on Rails, and Angular.js
- TensorFlow, the computational framework for building machine learning models
- Container technologies such as Docker and LXC
- Container orchestration frameworks such as Kubernetes, Docker Swarm, and Apache Mesos

Microservices

Overview

Microservices architectures structure an application as a suite of loosely coupled services that implement business capabilities. In contrast to a monolithic application, which requires developers to work sequentially on an application because the code is structured as a large, unitary assemblage, a microservices architecture empowers developers to swiftly update individual microservices in parallel. This ability to work on individual microservices in parallel accelerates development and enhances operational agility. For example, developers can initiate development-related work independently, without approval from a centralized IT governance authority, to work on microservices that are not the subject of other development efforts. Moreover, the separation of business functionalities into discrete microservices enables developers to expedite debugging efforts by rapidly zeroing in on the code that is responsible for the issue in question.

Individual microservices communicate with one another by means of application programming interfaces (APIs). APIs are central not only to communication between individual microservices but also to the consumption of microservices by end users. The importance of APIs to microservices applications means that developers of microservices-based applications need to ensure that APIs are collectively managed so that the individual microservices can optimally communicate with another. Technologies for managing APIs include API gateways, service mesh, and dedicated API management tools.

In a microservices model, the individual microservices that constitute an application can be independently deployed. In addition, the individual microservices may be developed in different languages and draw upon different storage technologies. Moreover, microservices architectures allow finely grained scaling of application components, whereas a monolithic application requires that the entire application scale. This ability to use finely grained, targeted scaling of application components optimizes application performance and enables cost savings resulting from optimized resource consumption.

Benefits

Open Source

Lower Costs

Open source software is typically free to download and is subsequently less costly than proprietary software. That said, many vendors choose to commercialize open source software by charging for software services and support as opposed to the software itself.

Transparency

One of the key benefits of open source technology is visibility into both the source code and its versioning history. This transparency empowers users to modify the source code or test its relevance for additional use cases and scenarios.

Security

Open source software is often thought to be more secure than proprietary software because anyone can view the original source code and identify security-related defects. Additionally, developers who leverage open source software can expediently add updates and modifications that render the source code even more stable and secure.

Reliability

An additional advantage of the transparency of open source software is increased reliability because developers enjoy streamlined opportunities to test the software in different hardware and software environments. Because developers can easily leverage open source software across a multitude of software stacks and deployments, they can proactively identify issues related to performance and compatibility and subsequently develop solutions that remediate them. As a result, open source software tends to be more reliable than proprietary software because of streamlined opportunities for it to be battle-tested in a variety of environments.

Longevity

The decentralized quality of open source software allows communities of practitioners to access the source code. As a result, open source software is thought to have greater longevity than proprietary software because it is less likely to be neglected or abandoned by a closed community of creators. In the event that the stewards of an open source software project are unable to maintain the software, the community can take responsibility for updating the software and ensuring its continued relevance to users.

Innovation

Open source technologies drive innovation because the decentralized quality of code contributions means that developers from a multitude of organizations, industry verticals, and geographies can contribute. Open source software inherently embraces contributions from large and diverse groups of developers; therefore, developers in an open source community are more likely to make innovative code contributions than developers in a closed source community, such as those within a specific enterprise.

Microservices

Ability to Deliver Superior Customer Experience and Business Innovation

Building superior customer experiences requires organizations to ship new features faster and keep iterating continuously. Microservices-based development empowers teams to quickly, iteratively, and automatically add new features to an application.

Accelerated Development

The microservices-based quality of cloud-native applications empowers development teams to work on individual microservices in parallel, thereby enabling developers to work on application development for multiple business functions concurrently.

Agility

Because applications are architected as microservices, developers can rapidly change components of an application without disrupting the rest of the application. This ability to change part of an application independently of other components underpins the implementation of a decentralized governance model that enables developers to make changes to an application without requiring the permission of other development teams.

Granular Application Scalability

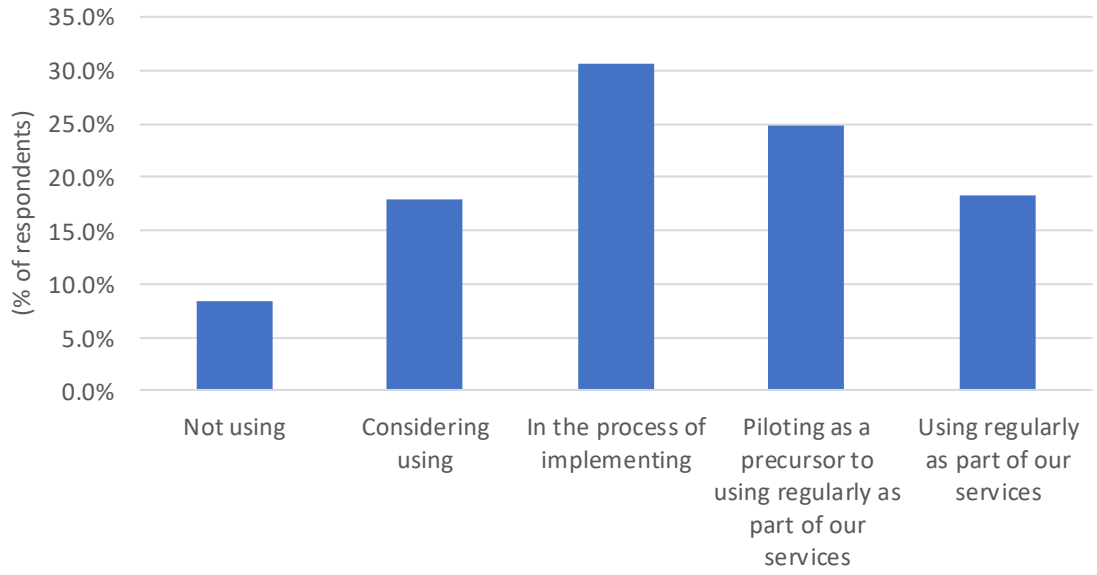
Developers can customize the scaling of individual microservices in contrast to unilaterally scaling the entire application. This ability to apply granular scaling control over constituent parts of an application optimizes scaling and enables more efficient resource consumption.

Trends

Open source technologies and microservices are two major trends in contemporary application development. Organizations are incorporating open source technologies into proprietary software as a means of using battle-tested solutions that have been subject to the scrutiny of the open source community. The use of open source software to develop proprietary solutions is known as "innersource" and is illustrative of the larger trend in contemporary application development toward code reusability and the use of code repositories such as GitHub. Examples of large-scale open source projects that organizations are incorporating into proprietary solutions include Kubernetes, TensorFlow, Terraform, and Golang. This trend toward incorporating open source solutions into proprietary software enables developers to harness the contributions of other developers. By reusing open source code, developers can increase development velocity, improve productivity, and focus their efforts on developing components of the digital solution in which they have dedicated expertise.

Microservices are central to the modernization of monolithic applications. Organizations are coming to terms with the need to modernize monolithic applications as a means of obtaining the capability to swiftly update their applications with new features and functionality to remain competitive with the digital transformation initiatives of their peers. In a recent IDC survey, respondents reflected on their organizations' adoption of microservices, as illustrated in Figure 1.

FIGURE 1: ***Organizational Adoption of Microservices***



Source: IDC's PaaSView and the Developer Survey, 2019

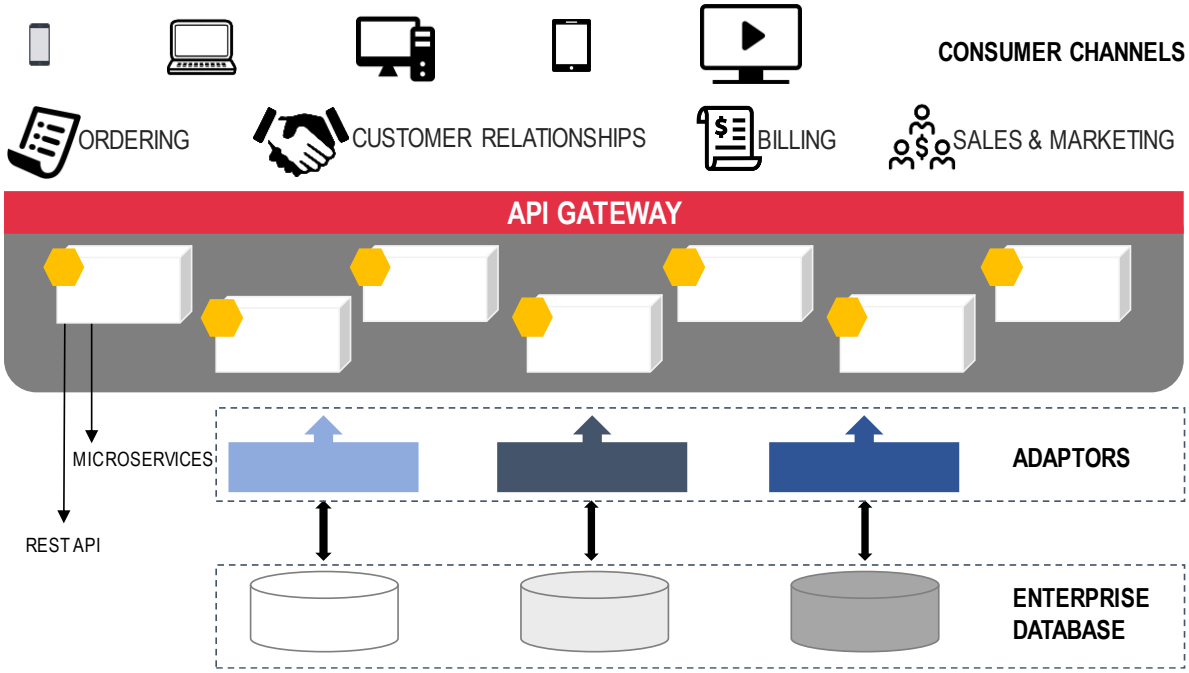
IDC's survey data shows that 73.4% of developers are considering using, implementing, or piloting microservices, while 18.2% of developers are actively using microservices. Furthermore, 23.8% of developers have used Spring Boot, an open source Java framework that developers use to create microservices, for at least 20% of applications they have developed in the past year. IDC's findings about microservices and Spring Boot adoption illustrate a strong growth forecast for the use of microservices worldwide.

Considering Tech Mahindra

Tech Mahindra's digital transformation accelerator, Blue Marble, helps enterprises digitize processes by leveraging their platform for modern application development (see Figure 2). The company is a global provider of IT services with about 121,000 employees across 90 countries and $4.9 billion in revenue. As enterprises embark on their journey to include automation in their business processes leveraging modern cloud infrastructure, digital transformation becomes an essential factor to ensure a successful outcome.

Additionally, TM Forum is a global industry initiative to collaboratively help communication service providers transform their business operations. Blue Marble makes available a repository of functional services that are compliant with TM Forum's Information Framework (SID) telecom data model, enabling organizations to quickly adopt industry standards to facilitate continued improvement of business processes.

FIGURE 2: ***High-Level Overview of Blue Marble***



Source: Tech Mahindra, 2019

Key Capabilities of Blue Marble

- Domain knowledge of telco challenges to understand existing systems and appropriately align solutions with customer needs
- Open source–based solution offering a wide array of community-driven technology that brings flexibility in assembling the most appropriate stack at a reduced cost

- Repository of reusable assets from TM Forum's SID telecom data model and TM Forum's Open APIs enabling quick deployment with reduced development effort
- Full-stack open architecture that is cloud agnostic
- Architecture built on standard data models and a single source of truth that enables customers to gain an omni-channel experience aligned with customer needs and behavior

Key Benefits of Blue Marble

- Optimization of resource utilization
- Continuous improvement in customer experience as feedback from channels layer is built into cycle times
- Software architecture rationalization taking monolithic applications and delivering them as microservices that can be independently built and managed, both functionally and infrastructurally
- Reduction of operational expenses (opex) by leveraging open source technologies
- Reduced time to market due to agility delivered by the solution after adopting modern application development approaches

Case Studies

United States–Based Tier 1 Telco — Application Modernization

This customer found that as a result of various acquisitions, mergers, and homegrown changes in existing commercial off-the-shelf (COTS) products, legacy systems had become inefficient due to task expansion beyond necessary functionalities. Two COTS products with the same functionality led to inconsistency and redundancy, and the applications did not have a single source of truth for their product-related data. This led to long turnaround times in releasing simple changes in the product and catalogue applications.

With Blue Marble, Tech Mahindra introduced a microservices-based business layer in between the system of engagement layer and the system of record layer. This decoupled the system of engagement layer from the legacy systems so that the customer experience is not a direct function of the dependencies and the vagaries of the existing enterprise systems. The microservices system created and deployed standardized SID-compliant services using an existing API gateway layer, setting up a continuous integration/continuous delivery (CI/CD) pipeline with on-premises deployment possible via Cloud Foundry.

The reported result was a 30–40% reduction in time to market while saving on licensing costs. According to the company, other benefits included infrastructure rationalization, unified customer experience, and agile delivery.

United States–Based Tier 1 Telco — Monolith Decomposition

This customer had a trouble and change management application that was a legacy monolith COTS product and had multiple geographic installations. Application maintenance, upgrades, and customization were major challenges plaguing the system. The customer also needed to negate scalability issues and long downtimes during the upgrades.
Tech Mahindra's Blue Marble solution provided domain decomposition of the application and translated that into

microservices that allowed the customer to upgrade at the component level rather than at the application level. The solution was deployed on client-specific cloud infrastructure.

The reported benefits included the complete use of open source tools and technologies, exempting the customer from licensing costs while allowing faster changes in the systems at lower cost and leading to significant reduction in downtime because of component-level upgrades. Features related to auto–ticket resolution using artificial intelligence/machine learning technologies are in the works.

Conclusion

IDC believes the market landscape for microservices-based development will grow significantly from 2019 to 2024. As such, solutions that accelerate and simplify the development of microservices-based applications will achieve greater importance, particularly as cloud adoption increases and more organizations seek to refactor monolithic and legacy applications for cloud-based deployment platforms using a microservices architecture. Just as the market landscape for microservices-based applications is expected to grow, so too is the landscape for open source technologies.

Open source technologies are especially important for microservices-based applications because they draw upon innovation that has been battle-tested by decentralized communities of users and communities spanning a wide range of industry verticals and geographies. Open source technologies are fundamental components of microservices-based applications at a multitude of layers of the technology stack. IDC expects a deepening of the integration between open source technologies and microservices-based development in forthcoming years in ways that accelerate development, decrease time to market, and reduce operational costs.

About the Analysts

Arnal Dayaratna, *Research Director, Software Development*

Dr. Arnal Dayaratna is Research Director, Software Development, at IDC. Dr. Dayaratna focuses on software developer demographics, modalities of software development, trends in programming languages and other application development tools, and the intersection of these development environments and the many emerging technologies that are enabling and driving digital transformation. Dr. Dayaratna's research examines how the changing nature of software development relates to broader trends in the technology landscape.

Larry Carvalho, *Research Director, Platform as a Service*

Larry Carvalho is Research Director for IDC's Platform as a Service (PaaS) practices. Mr. Carvalho focuses on cloud-enabled application development and directs research into the component competitive markets of cloud platforms and application services, including integration, analytics, application development, data management, IoT, and cloud testing.

IDC Custom Solutions

IDC Corporate USA

5 Speen Street
Framingham, MA 01701, USA

T 508.872.8200

F 508.935.4015

Twitter @IDC

idc-insights-community.com

www.idc.com