FORGEROCK

/WHITEPAPER

How to Implement Microservices within a Monolithic Architecture

Produced by:
Doug Norton-Bilsby, Global Vice President, Retail & Consumer Products, ForgeRock
Alex Laurie, Global VP, Solutions Architecture, ForgeRock

Table of Contents

Within today's disruptive economy, adaptability and agility are the most important organizational attributes to achieve long-term success. Additionally, due to the exponential increase in cyber-crime, embracing a zero trust model and having the ability to adopt new security features and practices quickly is a must for survival.

Of course, when it comes to system architectures, being adaptable, agile, and operating within a zero trust model is easier said than done. Most organizations' environments are built with business-critical monolithic legacy solutions that present significant challenges to adaptability, agility, and zero trust due to their siloed structures. Because of these challenges, leading organizations are transitioning to microservices for their customer facing applications within their current monolithic environments.

Implementing a microservices strategy promises transformative levels of agility, capability, and cost benefit. Yet, a microservices architecture introduces a new range of operational challenges and security risks. It is for this reason that identity is the enabler of the microservices movement.

Identity-enabled microservices delivered through a future-proof digital identity management platform increases security, efficiency, resiliency, and revenue. Additionally, it shows faster time-to-value, provides flexibility for changing requirements, and more easily supports mobile and IoT technologies.

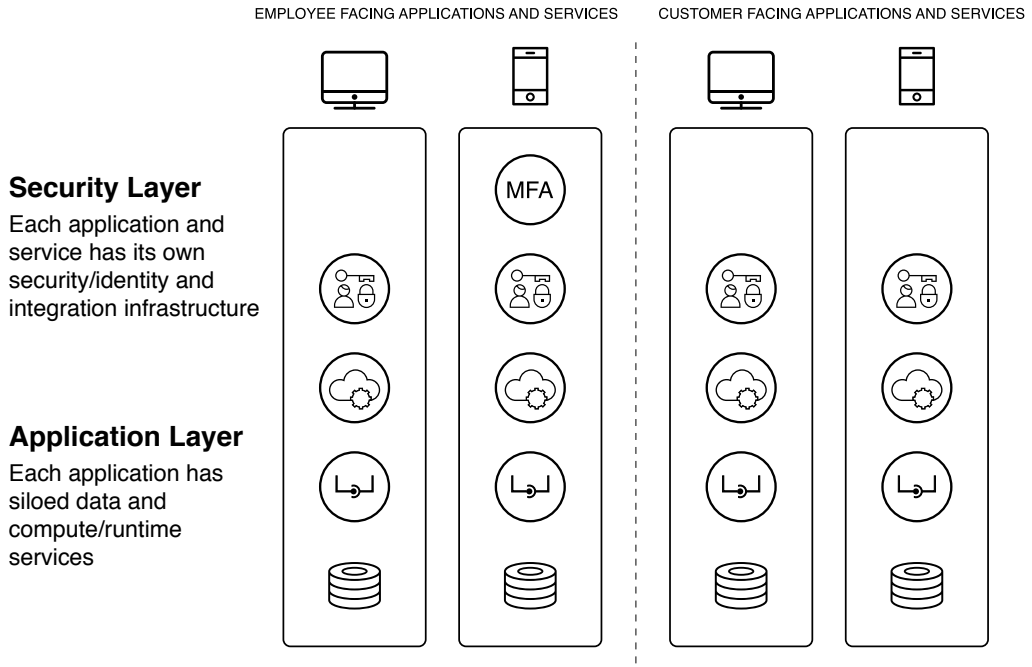
Monolithic Versus Microservices Architectures

Traditional applications and services are often considered 'monolithic' — that is, heavy technology solutions that require a lot of IT resources and money with the promise that the solutions will cover all bases for years to come.

Yet, as organizations move to a more agile-, data-, and API-driven world, these monolithic architectures do not cover the flexibility that organizations need to adapt to business and consumer requirements.

As diagram 1 illustrates, each application and service has siloed data, compute/runtime services, and storage. Additionally, each has its own identity, security, and integration infrastructure.

Diagram 1: A typical current state of enterprise application architecture



The inflexibility of monolithic architecture's siloed structure has been magnified in recent times by cloud services. As organizations are moving to and making larger investments in the cloud, those that are forward-thinking are trending towards microservices architectures.

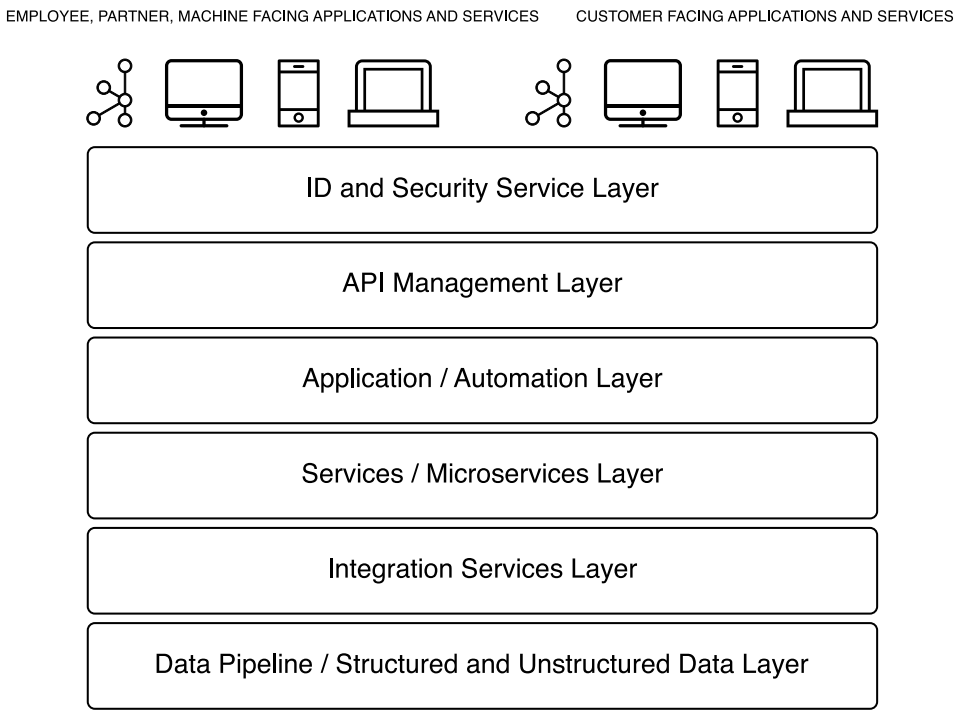
What are Microservices?

Microservices are based on an important development method that focuses on building and deploying applications as groups of modular, composable services within an application. Every aspect of the system is broken down to the smallest possible component. There are many advantages to a microservices approach, such as speed of development, resilience, speed of change, distribution of compute/bandwidth load, and reusability.

As a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services, microservices parallels development by enabling small autonomous teams to develop, deploy, and scale their respective services independently — meaning that developers can modify a service singularly without impacting other services. This makes building applications faster, easier, and capable of operating at an extremely high scale with the ability to change or evolve services in a much more agile way. From this, organizations can create a foundation that enables them to innovate and respond to customer needs more quickly.

If an organization has an existing monolithic architecture, it is still possible to leverage microservices. By breaking out identity and security (with identity) along with APIs, any organization can develop microservices to replace components of the monolithic architecture or leverage off-the-shelf services. For example, it is common to see that once the identity and integration platform are deployed, applications are taken out of the originating monolithic architecture and replaced by microservices based applications and third-party providers through API integrations within the integrations services layer, such as in diagram 2 below.

Diagram 2: The desired end state of enterprise application architecture



A core advantage of this approach is that it allows the business to change services when they find a better service or price from another third-party provider without having to change their entire technology stack. Development teams can also rapidly deliver highly agile microservice-based solutions without having to rip and replace whole systems — carefully putting the right technology in the right place.

As mentioned earlier, another advantage and benefit of microservices is reusability. Many systems need the same functions under the hood, so rather than rewrite the function for each system, developers can just point each system to the relevant microservice.

Leveraging Microservices within a Monolithic Architecture

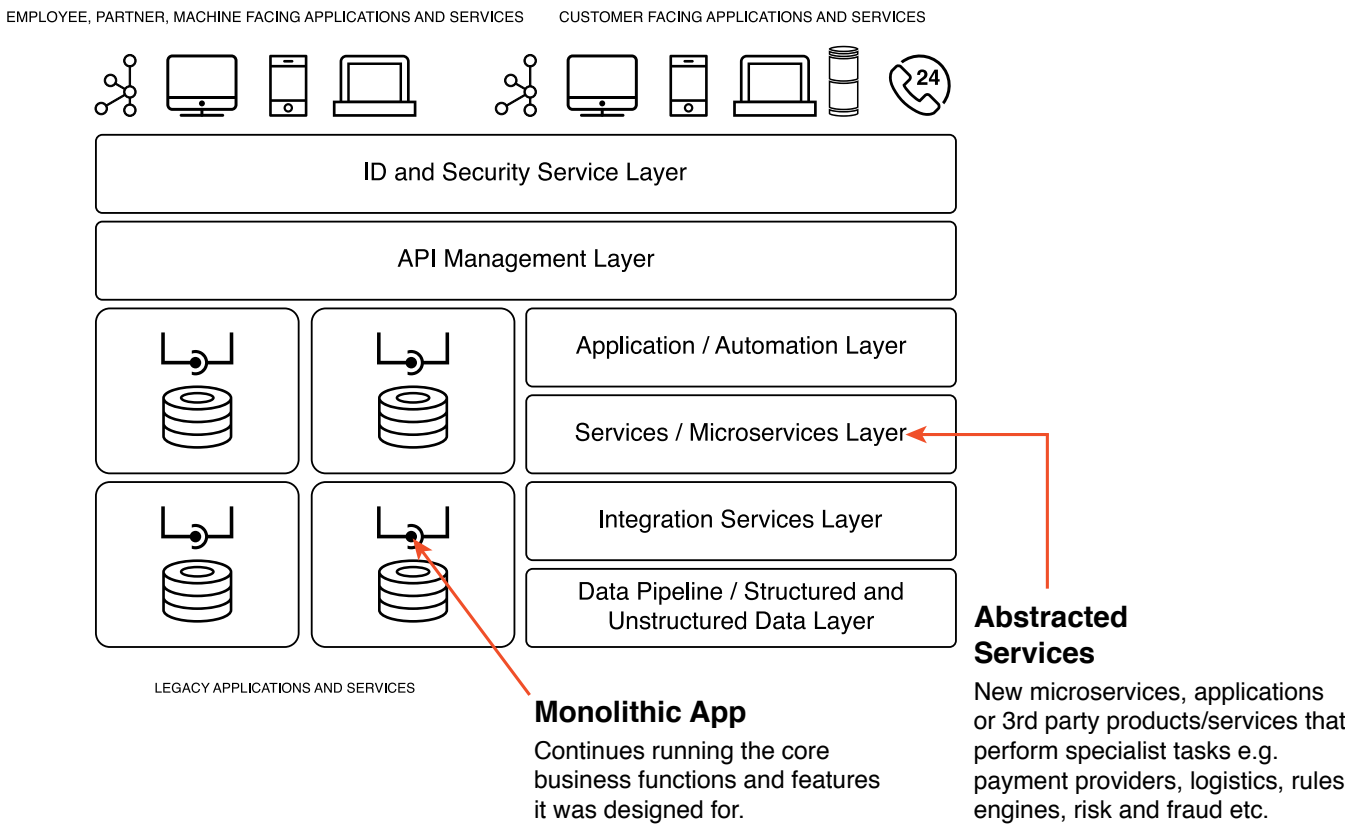
In order to leverage microservices within a monolithic architecture successfully, identity and security is vital. Identity and security, by way of an 'identity and security services layer', determines which service performs which function on behalf of which user within which business process. With identity-enabled microservices, not only does every transaction get protected, but each microservice as an identity itself is managed and secured.

In order to leverage microservices within a monolithic architecture, simply throwing out the old and replacing it with the new is not always possible nor desirable. Therefore, most organizations adopt a transitional approach of abstracting various applications and services using a combination of identity and APIs.

Building a single, back-end, identity and security service layer (see diagram 3) with identity for front-end applications to consume provides the security and functionality needed to achieve adaptability, agility, and zero trust. This identity and security service layer allows organizations to utilize third-party services as soon as they become available and provides a clear advantage by reducing overall software complexity.

An additional value to organizations is that identity, again as the identity and security service layer, enables services to map to a customer's preferred channels and understand the context of how they are used in order to add appropriate controls. For example, initiating multi-factor authentication using mobile when performing a bank transfer via Amazon's Alexa.

Diagram 3. Enabling a transition state architecture with identity and APIs



How It's Done: Transitioning to Microservices with Identity and API Integrations

As discussed so far, a combination of identity and API integrations provisioned as service layers over the top of monolithic systems enables new applications and services to integrate with monolithic systems. This structure maintains current back-end services while allowing the organization to create new customer journeys without having to spend huge amounts of effort and cost to change the legacy systems themselves. At the same time, it is also possible for organizations to add new automations, services, and data sources alongside these legacy systems to augment the end user experience/service as the coordination is abstracted from the legacy system. Over time, as legacy system business functionality is deprecated, replicated, or replaced by new services and technologies, the gradual sunsetting of the legacy systems is then enabled.

Identity Uplift and Abstraction

By abstracting back-end systems and services from channels and front-end services, organizations can take agile approaches to delivering new customer-facing applications and services, as well as take advantage of new channels, technologies, and data sources and tools when they become available.

When breaking down siloed, monolithic applications and services, a common issue is that each of these applications and/or services has its own identity management and storage capability. This means that user data is likely stored in a multitude of independent user credential stores — resulting in a fragmented view of the customer as well as limited cross-functional capabilities. To help with this, there are two identity service abstraction patterns:

Pattern 1: The identity service layer is configured to reach into each siloed application and run workflow to CRUD (Create, Read, Update, Delete) the user data. At the identity service layer, a common view of all the customer data is stored. This means, for example, that if a user updates their address in one system, the update can then be replicated across the other siloed systems. This is particularly important in terms of the ability to manage PII (Personally Identifiable Information) on behalf of customers for GDPR compliance.

Pattern 2: The identity service layer is configured to uplift* the identity data from each siloed application and store it in a centralized, highly available, and replicated identity store. With this approach, all aspects of identity management are abstracted from the siloed applications. This reduces the burden on siloed applications, and in some instances, also results in significant cost savings because it enables the sunsetting of systems and a more rapid integration of new systems/services to the ecosystem.

**The uplift can be configured so that it happens when the user signs into the system, thus making it a controlled switch-over.*

Application Deployment

Deploying an identity service along with API-based application integration separates internal and external facing applications from the back-end services, applications, data and integrations that support them. This separation allows the provision of a common set of identity, security, integration, and deployment standards that developer teams can leverage and deploy within their applications.

If an organization has a common identity service layer, developers can register a new application with the identity service and immediately start hitting the APIs in dev/test and through to production. They can also allow application developers to provide services to users without having to build new identity silos for each application. All of this means that a development team can build, integrate, and deploy applications much faster. Additionally, as applications and services are separated, developers can change the back-end services without having to re-develop front-end applications by simply adjusting the integration layer to point the right data at the right API.

Future-Proofing Investments

With any investment in technology or change in approach, a key goal for organizations is to future-proof themselves from exposure to technological, human, and legislative risks. Additionally, the sheer pace of change as well as ever-increasing volumes of data make it necessary for organizations to constantly adapt in order to maintain the necessary levels of service to users.

To meet the pace of change and future-proof in the areas of data, identity, and security, it is important to pick technologies that are standards based and supported, as well as have a commitment to maintaining standards within relevant industries. With standards-based supported technologies for identity, security, and API management (as well as data platforms, automation tooling, UI frameworks, and so on), organizations can be better prepared to address external factors such as legislation, changing technology, new ecosystems, and new security threats. Additionally, from the perspective

of developers rapidly deploying new technologies (internally or externally), having a standards-based approach along with a development framework enables much more flexibility.

Scale, Deployability, and Flexibility

The requirements for scale are only increasing as more consumers' primary interactions with organizations are digital. Additionally, the number of channels and devices that customers want to interact with organizations is also increasing faster than ever. For example, in 2017 the BBC stated that each user account on iPlayer had on average 2.5 devices* and that they expect this number to grow significantly — making the growth in total number of interactions between the organization and the customer exponential. For any significant organization, interacting with customers, devices, and IoT 'things' at scale requires a level of resilience and speed-of-response to match.

One of the perceived benefits of moving to the cloud, or cloud-style technologies, is the increased capacity for scale. Yet, if the architecture does not support horizontal cloud-style scaling and is not fully abstracted, organizations will still have to pay for more cloud resources (as well as people time) to make it scale on demand.

With the abstraction approach, organizations can focus scaling to the places that need it. If there are millions of user requests that typically peak on a certain day, it is possible to design the front-end applications to work at this scale and then build a back-end to support it. However, the applications, systems, and services deployed in this architecture also need to be designed for deployment and horizontal scaling within the paradigm of cloud technologies.

**The numbers were typical iPlayer-enabled devices, such as smart TVs and mobile devices, and did not include IoT and other devices such as Amazon Echo.*

ForgeRock Identity-Enabled Microservices

With identity as the key enabler, building a microservices architecture for front-end applications to consume within a monolithic architecture provides a clear advantage. By utilizing a single, identity solution, organizations can achieve adaptability, agility, and zero trust. Additionally, they can deliver on today and tomorrow's increasingly customer-centric, digital demands in a more secure, simple, and scalable way.

Identified as an Overall Leader by KuppingerCole, and the most visionary access management vendor by Gartner, ForgeRock provides the most flexible and unified microservices-ready identity solution.

ForgeRock's proven support for microservices architectures, and the new challenges they bring, accelerates and risk-mitigates investments. No other provider delivers the comprehensive capabilities to manage identity across an evolving microservices landscape. And, no other provider offers an end-to-end solution that checks all the boxes when migrating existing legacy applications to a microservices architecture.

With ForgeRock, organizations can simply solve complex problems, increase IT efficiency, and balance security and privacy to help meet demands and grow the business. ForgeRock does this by giving organizations the ability to:

- Utilize the best and most advanced authentication, authorization, and identity relationship management capabilities for all customer, employee and partner apps across the microservices landscape
- Abstract and utilize identity information from any monolithic applications to accelerate microservices builds from small to extremely complex
- Secure all microservices with a standards-based platform with modern token issuance and management
- Provide development and DevOps teams with the best digital identity management platform to reduce complexity
- Scale digital identity management to thousands of microservices and billions of identities (people, devices, things)

Learn More About ForgeRock-Enabled Microservices

With ForgeRock, organizations can easily build microservices within their current monolithic, legacy architectures in order to extend existing investments, streamline operations, and grow business. Contact us today to learn more.

/ABOUT FORGEROCK

ForgeRock®, the leader in digital identity management, transforms how organizations build trusted relationships with people, services, and things. Monetize customer relationships, address stringent regulations for privacy and consent (GDPR, HIPAA, Open Banking, etc.), and leverage the internet of things with ForgeRock. We serve hundreds of brands, including Morningstar, Vodafone, GEICO, Toyota, and Pearson, as well as governments like Norway and Canada. www.forgerock.com