IBM Cloud
Technical white paper

Agile Integration Architecture

Container-Based and Microservices-Aligned Lightweight Integration Runtimes

Contents

1 Executive summary

2 Integration Has Changed

2 The Journey So Far – SOA, ESBs and APIs

3 The Case for Agile Integration Architecture

3 Aspect 1: Fine-grained Integration Deployment

4 Aspect 2: Decentralized integration ownership

5 Aspect 3: Cloud-native integration infrastructuren

5 How has the modern integration runtime changed to accommodate agile integration architecture?

6 Agile Integration Architecture for the Integration Platform

11 The IBM Cloud Integration Platformn

Embrace digital transformation with agile integration centered around an equally agile approach, giving you the ability to move quickly to meet the demands of multicloud, decentralization and microservices.

Executive summary

Organizations pursuing digital transformation must embrace new ways to use and deploy integration technologies, so they can move quickly in a manner appropriate to the goals of multicloud, decentralization and microservices. The application integration layer must transform to allow organizations to move boldly in building new customer experiences, rather than forcing models for architecture and development that pull away from maximizing the organization's productivity.

Many organizations have started embracing agile application techniques such as microservices architecture and are now starting to see the benefits of that shift. This approach compliments and accelerates an enterprise's API strategy. Businesses should also seek to use this approach to modernize their existing ESB infrastructure to achieve more effective ways to manage and operate their integration services in their private or public cloud.

This white paper is derived from a book that explores the merits of what we refer to as **agile integration architecture**-a container-based, decentralized and microservices - aligned approach for integration solutions that meets the demands of agility, scalability and resilience required by digital transformation.

Integration Has Changed

IDC estimates that spending on digital transformation initiatives will represent a $20 trillion market opportunity over the next 5 years[1].What's behind this staggering explosion of spending? The ever-present, ever-growing need to build new customer experiences through connected experiences across a network of applications that leverage data of all types.

That's no easy task – bringing together processes and information sources at the right time and in the right context is difficult at best, particularly when you consider the aggressive adoption of SaaS business applications. New data sources need to be injected into business processes to create competitive differentiation.

"To drive new customer experiences organizations must tap into an ever-growing set of applications, processes and information sources – all which significantly expand the enterprise's need for and investment in integration capabilities."

The Value of Application Integration for Digital Transformation

When you consider your agenda for building new customer experiences and focus on how data is accessed and made available for the services and APIs that power these initiatives, you can see several significant benefits that application integration brings to the table:

- Effectively addressing disparity – Access data from any system in any format and build homogeneity from it, no matter how diverse your multicloud landscape grows.
- Expertise of the endpoints – Modern integration includes smarts around complex protocols and data formats, but it also incorporates intelligence about the actual objects, business and functions within the end systems.
- Innovation through data – Applications owe much of their innovation to their opportunity to combine data beyond their boundaries and create meaning from it, a trait particularly visible in microservices architecture.
- Enterprise-grade artifacts – Integration flows inherit a tremendous amount of value from the runtime, which includes enterprise-grade features for error recovery, fault tolerance, log capture, performance analysis, and much more.

The integration landscape is changing to keep up with enterprise and marketplace computing demands, but how did we get from SOA and ESBs to modern, containerized, agile integration architecture?

The Journey so Far – SOA, ESBs and APIs

Before we can look forward to the future of agile integration, we need to understand what came before. SOA (Service Oriented Architecture) patterns emerged at the start of the millennium, and at first the wide acceptance of the standards SOA was built upon heralded a bright future where every system could discover and talk to any other system via synchronous exposure patterns.

Fast forward a bit and you will find yourself right in the middle of the ESB (Enterprise Service Bus) movement – a technology that was supposed to provide connectivity to backend systems, coming from the preceding hub-and-spoke pattern. While many enterprises successfully implemented the ESB pattern, the term isn't exactly feeling the love from cloud-native space. It's seen as heavyweight and lacking in agility. How did we go from one extreme to another?

The truth boils down to a few, often interrelated, factors:

- SOA was more complex than just the implementation of an ESB, particularly around who would fund an enterprise-wide program.
- ESB patterns formed a single infrastructure for the whole enterprise, with tens or hundreds of integrations installed on a production server cluster. Although heavy centralization isn't required by the ESB pattern, the resulting topologies almost always fell prey to it.

[1]IDC MaturityScape Benchmark: Digital Transformation Worldwide, 2017, Shawn Fitzgerald. Golluscio.

- Centralized ESB patterns often failed to deliver the significant savings companies were hoping for, since interfaces could not be re-used from one project to another.
- Cross-enterprise initiatives like ESB struggled to find funding, and often that funding only applied to services that would be reusable enough to cover their creation cost.

ESB patterns have had issues ensuring continued funding for cross-enterprise initiatives since those do not apply specifically within the context of a business initiative.

The result was that creation of services by this specialist SOA team became a bottleneck for projects rather than the enabler that it was intended to be. This typically gave the centralized ESB pattern a bad name by association.

Service-oriented architecture as applied to ESB patterns is an enterprise-wide initiative to create re-usable, synchronously available services and APIs, such that new applications can be created more quickly incorporating data from other systems.

Microservices architecture, on the other hand, is an option for how you might choose to write an individual application in a way that makes that application more agile, scalable, and resilient.

The Case for Agile Integration Architecture

Why have microservices concepts become so popular in the application space? They represent an alternative approach to structuring applications. Rather than an application being a large silo of code running on the same server, the application is designed as a collection of smaller, completely independently-running components.

Microservices architecture enables three critical benefits:

1. Greater **agility** – Microservices are small enough to be understood completely in isolation and changed independently.
2. Elastic **scalability** – Their resource usage can be fully tied into the business model.
3. Discrete **resilience** – With suitable decoupling, changes to one microservice do not affect others at runtime.

With those benefits in mind, what would it look like if we reimagined integration, which is typically deployed in centralized silos, with a new perspective based on microservices architecture? That's what we call **"agile integration architecture."**

Agile integration architecture is defined as "a container-based, decentralized and microservices-aligned architecture for integration solutions."

There are three related, but separate aspects to agile integration architecture:

Aspect 1: Fine-grained integration deployment.
What might we gain by breaking out the integrations in the siloed ESB into separate runtimes?

Aspect 2: Decentralized integration ownership.
How should we adjust the organizational structure to better leverage a more fine-grained approach?

Aspect 3: Cloud native integration infrastructure.
What further benefits could we gain by a fully cloud-native approach to integration.

Aspect 1: Fine-grained Integration Deployment

The centralized deployment of integration hub or ESB patterns where all integrations are deployed to a single heavily nurtured (HA) pair of integration servers has been shown to introduce a bottleneck for projects. Any deployment to the shared servers runs the risk of destabilizing existing critical interfaces. No individual project can choose to upgrade the version of the integration middleware to gain access to new features.

We could break up the enterprise-wide ESB component into smaller, more manageable and dedicated pieces. Perhaps in some cases we can even get down to one runtime for each interface we expose. These “fine-grained integration deployment” patterns provide specialized, right-sized containers, offering improved agility, scalability and resilience, and look very different to the centralized ESB patterns of the past. Figure1 demonstrates in simple terms how a centralized ESB differs from fine-grained integration deployment.

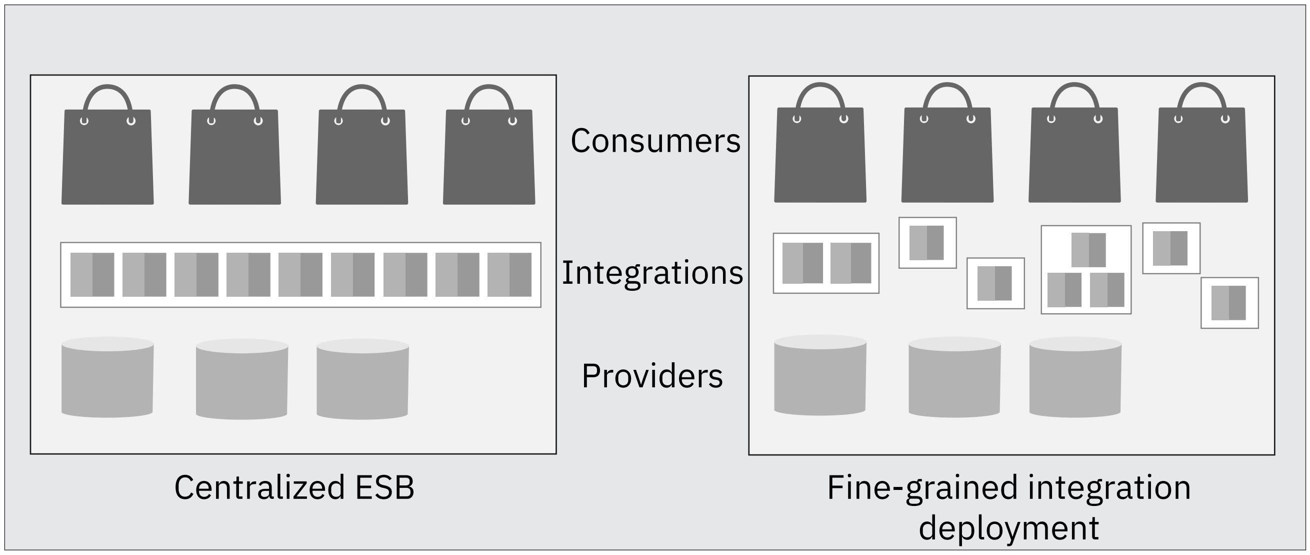

Figure 1: Simplified comparison of a centralized ESB to fine-grained integration deployment

Fine-grained integration deployment draws on the benefits of a microservices architecture. Let’s revisit what we listed as microservices benefits in light of fine-grained integration deployment:

- **Agility** – Different teams can work on integrations independently without deferring to a centralized group or infrastructure that can quickly become a bottleneck. Individual integration flows can be changed, rebuilt, and deployed independently of other flows, enabling safer application of changes and maximizing speed to production.
- **Scalability** – Individual flows can be scaled on their own, allowing you to take advantage of efficient elastic scaling of cloud infrastructures.
- **Resilience** – Isolated integration flows that are deployed in separate containers cannot affect one another by stealing shared resources, such as memory, connections, or CPU.

When you think about agility, scalability and resilience, it’s important to remember that you can’t achieve these benefits of fine-grained integration without decentralized integration.

Learn so much more about fine-grained integration in our book Agile Infrastructure Architecture, available now for download!

Aspect 2: Decentralized integration ownership

A significant challenge faced by service-oriented architecture was the way that it tended to force the creation of central integration teams, and infrastructure to create the service layer.

This created ongoing friction in the pace at which projects could run since they always had the central integration team as a dependency. The central team knew their integration technology well, but often didn’t understand the applications they were integrating, so translating requirements could be slow and error prone.

Many organizations would have preferred the application teams own the creation of their own services, but the technology and infrastructure of the time didn't enable that.

The move to fine-grained integration deployment opens a door such that ownership of the creation and maintenance of integrations can be distributed. It's not unreasonable for business application teams to take on integration work, streamlining the implementation of new capabilities.

Piqued your curiosity about fine-grained integration deployment? Answer your question with our Agile Infrastructure Architecture book, available now!

Aspect 3: Cloud-native integration infrastructure

Integration runtimes have changed dramatically in recent years. So much so that these lightweight runtimes can be used in truly cloud-native ways. By this we are referring to their ability to hand off the burden of many of their previously proprietary mechanisms for cluster management, scaling, availability and to the cloud platform in which they are running.

This entails a lot more than just running them in a containerized environment. It means they have to be able to function as "cattle not pets," making best use of the orchestration capabilities such as Kubernetes and many other common cloud standard frameworks.

Adopting a "cattle approach" impacts the ways in which your DevOps teams will interact with the environment and the solution overall, create increasing efficiencies as more solutions are moved to lightweight architectures.

How has the modern integration runtime changed to accommodate agile integration architecture?

Clearly, agile integration architecture requires that the integration topology be deployed very differently. A key aspect of that is a modern integration runtime that can be run in a container-based environment and is well suited to cloud-native deployment techniques. Modern integration runtimes are almost unrecognizable from their historical peers. Let's have a look at some of those differences:

- **Fast lightweight runtime:** They run in containers such as Docker and are sufficiently lightweight that they can be started and stopped in seconds and can be easily administered by orchestration frameworks such as Kubernetes.
- **Dependency free:** They no longer require databases or message queues, although obviously, they are very adept at connecting to them if they need to.
- **File system based installation:** They can be installed simply by laying their binaries out on a file system and starting them up–ideal for the layered file systems of Docker images.
- **DevOps tooling support:** The runtime should be continuous integration and deployment–ready. Script and property file-based install, build, deploy, and configuration to enable "infrastructure as code" practices. Template scripts for standard build and deploy tools should be provided to accelerate inclusion into DevOps pipelines.
- **API-first:** The primary communication protocol should be RESTful APIs. Exposing integrations as RESTful APIs should be trivial and based upon common conventions such as the Open API specification. Calling downstream RESTful APis should be equally trivial, including discovery via definition files.
- **Digital connectivity:** In addition to the rich enterprise connectivity that has always been provided by integration runtimes, they must also connect to modern resources. For example, NoSQL databases (MongoDb and Cloudant etc.), and messaging services such as Kafka. Furthermore, they need access to a rich catalogue of application intelligent connectors for SaaS (software as a service) applications such as Salesforce.

- **Continuous delivery:** Continuous delivery is enabled by command-line interfaces and template scripts that mesh into standard DevOps pipeline tools. This further reduces the knowledge required to implement interfaces and increases the pace of delivery.
- **Enhanced tooling:** Enhanced tooling for integration means most interfaces can be built by configuration alone, often by individuals with no integration background. With the addition of templates for common integration patterns, integration best practices are burned into the tooling, further simplifying the tasks. Deep integration specialists are less often required, and some integration can potentially be taken on by application teams as we will see in the next section on decentralized integration.

Modern integration runtimes are well suited to the three aspects of agile integration architecture: fine-grained deployment, decentralized ownership, and true cloud-native infrastructure.

Want an even deeper dive into cloud-native infrastructure? Download our Agile Integration Architecture book now!

Agile Integration Architecture for the Integration Platform

Throughout this paper, we have been focused on the application integration features as deployed in an agile integration architecture. However, many enterprise solutions can only be solved by applying several critical integration capabilities. An integration platform (or what some analysts refer to as a "hybrid integration platform") brings together these capabilities so that organizations can build business solutions in a more efficient and consistent way.

Many industry specialists agree on the value of this integration platform. Gartner notes:

The hybrid integration platform (HIP) is a framework of on-premises and cloud-based integration and governance capabilities that enables differently skilled personas (integration specialists and nonspecialists) to support a wide range of integration use cases.… Application leaders responsible for integration should leverage the HIP capabilities framework to modernize their integration strategies and infrastructure, so they can address the emerging use cases for digital business[2].

One of the key things that Gartner notes is that the integration platform allows multiple people from across the organization to work in user experiences that best fits their needs. This means that business users can be productive in a simpler experience that guides them through solving straightforward problems, while IT specialists have expert levels of control to deal with the more complex enterprise scenarios. These users can then work together through reuse of the assets that have been shared; while preserving governance across the whole.

Satisfying the emerging use cases of the digital transformation is as important as supporting the various user communities. The bulk of this paper will explore these emerging use cases, but first we should further elaborate on the key capabilities that must be part of the integration platform.

The IBM Cloud Integration Platform

IBM Cloud Integration brings together the key set of integration capabilities into a coherent platform that is simple, fast and trusted. It allows you to easily build powerful integrations and APIs in minutes, provides leading performance and scalability, and offers unmatched end-to-end capabilities with enterprise-grade security.

Within the IBM Cloud Integration platform, we have coupled the six key integration specialties each a best-of-breed feature. These are:

API Management:
Exposes and manages business services as reusable APIs for select developer communities both internal and external to your organization. Organizations adopt an API strategy to accelerate how effectively they can share their unique data and services assets to then fuel new applications and new business opportunities.

Security Gateway:
Extend Connectivity and Integration beyond the enterprise with DMZ-ready edge capabilities that protect APIs, the data they move, and the systems behind them

Application Integration:
Connects applications and data sources on-premises or in the cloud, in order to coordinate exchange business information so that data is available when and where needed.

[2]Hype Cycle for Application Infrastructure and Integration, 2017, Elizabeth

Messaging:
Ensures real-time information is available from anywhere at anytime by providing reliable message delivery without message loss, duplication or complex recovery in the event of system or network issue.

Data Integration:
Accesses, cleanses and prepares data to create a consistent view of your business within a data warehouse or data lake for the purposes of analytics.

High Speed Transfer:
Move huge amounts of data between on-premises and cloud or cloud-to-cloud rapidly and predictably with enhanced levels of security. Facilitates how quickly organizations can adopt cloud platforms when data is very large.

Through this teaser white paper, hopefully you've gotten a broader perspective of the various critical capabilities required as part of an integration platform, a sense of the requirements for those capabilities to work together, and an appreciation of how the agile integration architecture can be adopted to enable greater agility, scalability and resilience for the platform.

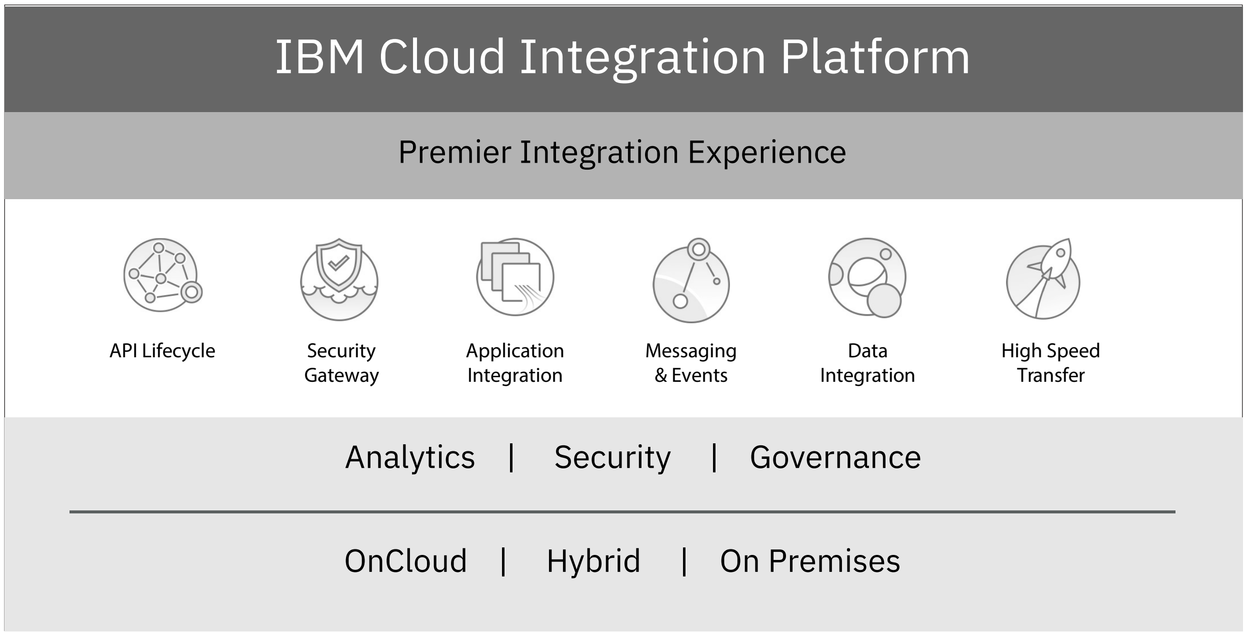
IBM Cloud Integration Platform
Premier Integration Experience
API Lifecycle
Security Gateway
Application Integration
Messaging & Events
Data Integration
High Speed Transfer
Analytics | Security | Governance
OnCloud | Hybrid | On Premises

Figure 2: The IBM Cloud Integration Platform

Make sure to download the comprehensive e-book to learn even more about agile integration architecture.