O'REILLY®

Integration and the Path to Becoming a Digital Business

Why and How to Interconnect Everything

Ciara Byrne

TIBCO®

INTERCONNECT

Solve the Pervasive Need for Integration

Integration using APIs, microservices, and event-driven architecture patterns is the connective tissue that binds modern application services.

Learn how TIBCO's integration solutions fuel digital business transformation with:

- Cloud PaaS and serverless deployments, including AWS Lambda
- New application development methods like microservices and events
- Machine learning intelligence for IoT and edge computing

tibco.com/interconnect-everything

CONNECTED
INTELLIGENCE

Integration and the Path to Becoming a Digital Business

Ciara Byrne

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Integration and the Path to Becoming a Digital Business
by Ciara Byrne

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Rachel Roumeliotis
Production Editor: Justin Billing
Copyeditor: Octal Publishing
Proofreader: Amanda Kersey
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

November 2017: First Edition

Revision History for the First Edition
2017-11-16: First Release

978-1-492-02529-0

[LSI]

Table of Contents

Integration and the Path to Becoming a Digital Business

Every company is now a software company. Digital transformation allows even large enterprises to adapt to changes in markets and customers at lightning speed, responding with new products, new processes, and new business models. After decades of outsourcing IT, every enterprise must now solve complex business problems with software. Digital transformation doesn't just require new technology; it requires a new, more agile mindset. Every line of business must have access to digital tools needed to innovate at the edge, and it's the job of the core IT team to provide them.

Digital transformation relies on connecting data and systems, people and processes. Integration technologies have traditionally formed the nervous system of a large enterprise, connecting systems and moving data. But the human nervous system doesn't just connect and sense; it also acts on data in real time. A digital business technology platform augments the intelligence of a digital business by building on its ability to connect and sense, to learn and act automatically, and enables the next stage of your digital transformation.

Unlike a biological nervous system, an enterprise's digital business technology platform will reach beyond the traditional boundaries of the business to run in the cloud or on devices within the Internet of Things (IoT). A digital business platform will build on traditional integration technologies, extending them to deal with microservices, serverless architectures, event-driven architectures, machine learning, and edge intelligence. Agile development methodologies, DevOps processes, bimodal IT, and other cultural changes are pieces in a digital business jigsaw. This report is your guide to creating a digital platform in your company.

Why You Need a Digital Business Platform

A digital business platform helps businesses identify opportunities and solve problems by connecting, sensing, learning, and automating. Gartner predicts that by 2020 persuasive algorithms capturing insights from psychology and cognitive science will alter the behavior of one billion employees, and consumers will have more conversations with bots than with their spouses. The robotic enterprise will automate entire swathes of business processes. Augmenting and automating the intelligence of the enterprise is crucial to the future of business.

Every industry sector faces different business challenges, and will therefore emphasize different aspects of a digital business platform. "I think it's fair to characterize that there's two flavors of a digital business platform," says Rajeev Kozhikkattuthodi, vice president of product management at TIBCO Software. "One is more around edge and augmenting intelligence in real time. The other flavor is more focused on digital transformation as fundamentally enabling your relationship with customers and partners."

Manufacturing, transportation, utilities, and other industries with large capital investments need to maximize the value of their infrastructure and improve operational efficiency. Adding intelligence at the edge so that IoT devices can sense and act locally will be an essential component of a digital business platform in these sectors.

Banks and telecommunications companies want to redefine their relationship with their customers, creating the kind of seamless, omnichannel environments at which digital retailers have excelled. Personalizing interactions with customers will require digital businesses to combine machine learning models with real-time streaming data.

Retailers and travel companies, which must provide both a personalized customer experience and efficient delivery or transport with physical infrastructure, face both of these challenges.

Every enterprise wants to reduce costs and increase the speed of innovation. Every enterprise needs to make faster and better decisions with machine learning and real-time analytics. A robust digital business platform can support these goals.

Culture and Digital Transformation

The digital transformation of a large enterprise is a daunting task, and cultural and organizational change can be its most challenging aspects. Digital enterprises value speed and experimentation, embrace risk, and are distributed rather than hierarchical.

"Personally, I think that the technology pieces are easier to solve," says Rahul Kamdar, director of product management and strategy at TIBCO. "Culture is a tougher one. When I say culture, it's really about changing how you structure your team's accountability and how people work with each other."

One of the key obstacles to digital transformation is putting too much emphasis on technology, and not enough on the work processes that the technology enables. The basic organizational changes required for digital transformation like creating cross-functional product teams to develop business capabilities and implementing Agile development and DevOps processes are essential prerequisites to creating a digital business platform.

Digital transformation requires total organizational commitment. Conway's law indicates that organizations produce software that reflects the structure of the organization itself.

Core Versus Edge IT

Much of the innovation in a digital enterprise will happen at the edge, and this trend will only accelerate with the addition of a digital business platform that extends beyond traditional enterprise boundaries. Lines of business must have the freedom to experiment and to build their own services and applications. Decision-making must become distributed rather than centralized.

"The reality is, especially if you're an incumbent, somebody out there is out-innovating you with a much-reduced footprint and a deeper appetite for experimentation," says Kozhikkattuthodi. "So, product culture, data-driven culture, experimentation, all of these things are different facets of what truly is the social-technical landscape of digital transformation."

These requirements are drastically changing the role of the core IT team. Mission-critical core business processes will still be built and run by core IT. However, the core IT team is increasingly an enabler of innovation within the lines of business. For example, core IT teams are responsible for sourcing, evaluating, and integrating software as a service (SaaS) apps into their existing operations and business processes.

Digital initiatives at the edge need access to data and services that reside in datacenters, which core IT will typically expose in the form of APIs. Line-of-business developers will create applications from internal and external services and legacy code, glued together with APIs. Core IT can also enable business users to become citizen integrators by giving them self-service tools to integrate data and applications.

In summary, core IT will focus on trust, governance, and reusability, whereas edge IT will focus on speed, agility, and experimentation. Gartner calls this bimodal IT: mode 1 provides stable and reliable performance to address the daily business functioning of the enterprise, whereas mode 2 emphasizes flexibility and responsiveness to drive new business outcomes. Both modes must operate cooperatively to achieve enterprise business objectives.

A Shared Cloud Governance Team

Core IT still needs to exert some control over cloud services and how they are used. SaaS tools should be vetted for security, for example, before they are made available within the enterprise. Recent Skyhigh research shows that an average company uses more than 1,159 cloud services, most of which are shadow cloud services whose usage IT has not sanctioned.

Core IT teams might also provide self-service cloud computing infrastructure running on a private cloud to internal developers. Users can request or configure servers and launch applications in a completely automated fashion. Common data access and security procedures should be defined for shared cloud infrastructure.

Cloud governance is a new task in most enterprises. A survey by Cloud Security Alliance (CSA) showed that only 21% of companies have a cloud governance team, and only 16% actually had an acceptable cloud usage policy. The shared cloud governance team should be staffed by stakeholders from around the company but is typically led by an enterprise architect team with cloud security and compliance skills. The cloud governance team is not a gatekeeper in the traditional sense; rather, it is an internal consulting team with enforcement abilities whose objective is to allow edge teams to be as independent as possible within policy parameters.

The cloud governance team often also defines the company API style guide and API policies. All internal developers should adhere to the style guide even when transforming an existing service or importing an API from an external tool.

A Hybrid Integration Platform for Digital Transformation

Gartner has defined pervasive integration as integrating on-premises and cloud applications and data sources, business partners, clients, mobile apps, social networks, and IoT devices to enable organizations to pursue digital business. The foundation of a digital business platform is a hybrid integration platform (HIP) which meets the organization's pervasive integration needs by supporting different types of users, from integration specialists to business users, and diverse deployment models: on-premises, cloud, mobile, and edge devices for the IoT.

Microservices, Containers, and APIs

Microservices, containers, and APIs are the holy trinity of digital transformation technologies and as such are central to pervasive integration. Microservices structure an application as a suite of small, loosely coupled, collaborating business services that can be written in different languages and deployed independently. Many SaaS services can also be considered microservices. The microservices architecture enables the continuous delivery and deployment of large, complex applications. Integration becomes more important than ever in a microservices world in which small, specialized services must be orchestrated to achieve business goals.

Microservices are often exposed via APIs and deployed in containers. A container includes everything needed to run a microservice—binaries, libraries, configurations—bundled into one lightweight, portable package. Containers allow microservices to reliably and seamlessly move from one computing environment to another, whether that's a developer's laptop or a public cloud. Developers can build once and deploy anywhere without code changes.

Microservices, containers, and APIs support digital transformation by letting developers build and modify services faster and more flexibly, and automatically deploy them to public, private, or hybrid clouds or at the edge. Hybrid integration in a microservices world must cover a wide spectrum of use cases from core to edge.

Core Integration Needs

Core IT teams will build custom cloud-native (or a mix of cloud and on-premises) applications with an API-first approach and deploy them to container-based platforms. Core IT will also provide lines of business with core data and services in the form of microservices exposed via APIs.

APIs are the nerves of the digital business nervous system. Core IT teams need full life cycle API management to model, create, publish, operate, and maintain internal and external APIs. The API management system should also enforce the API policies, such as security requirements or throttling service-level agreements (SLAs), defined by the shared cloud governance team.

Integration platform as a service (iPaaS) are integration solutions for a cloud-native world. An iPaaS enables the development, execution and governance of complex integration flows connecting any combination of on-premises and cloud-based microservices, SaaS tools, and data. Because an iPaaS is entirely hosted and managed by an integration vendor, developers can seamlessly connect to cloud and on-premises services without having to worry about the underlying infrastructure. The iPaaS should, however, enforce the cloud governance policies defined by the enterprise's shared cloud governance team.

Edge Integration Needs

Application platform as a service (aPaaS) is a low-code application environment that lets citizen integrators (who might be line-of-business developers or business users) build fully functional applications. An aPaaS might also support integration use cases in which users can take advantage of prebuilt connectors, connect apps, or pull information together from core systems.

Citizen integrators will create new integration flows by configuring them rather than building them from scratch. Integration service as a service (iSaaS) products allow business users with no coding skills to integrate data between various cloud services. Business users might not think of this as integration at all; they simply want to share data between systems without copying it manually.

IoT use cases introduce new integration requirements. An IoT integration gateway must interconnect all edge devices using a variety of IoT communication standards. IoT devices generate large volumes of sensor data, and IoT gateways might have unreliable, expensive, or slow connectivity to central or cloud services. IoT services, therefore, cannot always rely on cloud processing to make real-time decisions. Instead, gateways can process some data and make some decisions locally in real time. An IoT integration gateway can filter and aggregate sensor data streams at the edge and send only relevant information such as errors or alerts back to a private datacenter or public cloud.

API Management

An API is no longer just a technical interface. APIs are the on-ramp to digital transformation. As such, an organization's digital strategy should drive its API program and determine the types of APIs that it builds.

Internal API programs can help you become more agile because it's faster to build new services by reusing existing services exposed by APIs. Internal APIs also make it easier for line-of-business developers and citizen integrators to build innovative edge services in low-code or no-code environments.

Some APIs become revenue-generating products. Amazon Web Services (AWS) is one result of Amazon CEO Jeff Bezos' 2002 API mandate, in which all Amazon teams were instructed to expose their data and functionality through APIs. In 2016, AWS earned Amazon $12.2 billion.

"The point of that memo was that it created a culture of agility," says Robert Zazueta, director of digital strategy at TIBCO. "Everything that we create needs to be thought about as potentially reusable. Even the folks who were building the architecture that ran Amazon's stores, they had to build an API layer."

Partner API programs can make existing processes such as sharing key business data more efficient. Comcast's partner API program, for example, allows secure data exchanges between its partners and their customers, and the time required to onboard partners was cut from four weeks to just a few hours. Cisco's nearly 70,000 channel partners worldwide account for more than 80 percent of the company's revenue, so the company's support API program for partners is essential to its business.

Public APIs exposing company data or services to third-party developers extend innovation beyond the boundaries of the enterprise itself and can also become new sources of revenue.

APIs Are Products

Taking an API-first approach leads to the creation of internal and external APIs, APIs that expose data and business functionality, and APIs that orchestrate other APIs. Due to the granularity of microservices, the number of APIs that most enterprises manage will only increase. It's not enough to create some APIs and hope for the best. A successful API program must be managed to ensure that it serves the goals of the business. APIs are products, and the developers who use your APIs, whether they are internal or external, should be treated as customers.

"You have to approach your internal developers in the exact same way you would external developers with good documentation, a good developer experience, good internal support, and some level of evangelism," says Zazueta. "Put someone in there who sits in every single meeting and hears every potential problem for every potential project coming up and says the API can do that. That's critical."

Monetization of external APIs, or other methods of creating value, like generating new business opportunities via APIs, must be thought through in order to provide sustainable incentives and rewards both to the API provider and to the developer. At a minimum the cost of delivery of the API should be covered by the value created. Building unused APIs is a waste of IT time and resources.

Key performance indicators (KPIs) should be defined for each API. API KPIs linked to user metrics—such as registered API keys, stickiness, and usage trends, and operations metrics such as availability, response time, queries per second, or error rates—are often used. However, the most important KPIs link API calls back to a business goal like revenue or cost savings. All of this means that APIs need product managers.

"It's critical to have a dedicated API team, even if that team is literally one person who is the center point and the enforcer for the [API] style guide," says Zazueta. "Somebody who knows the purpose of the API, the design of the API, handles the code reviews, all that kind of stuff."

Enterprises should provide APIs that are enterprise-grade, that are secure and scalable, and that provide business value. That makes API management tools essential. An API management platform generally provides a developer portal to assist and govern the communities of developers using your APIs, an API registry where data and functionality is actually stored, and an API gateway, which processes API calls, filters traffic, and enforces API policies. Scalability, security and support, and analytics are also handled by API management platforms. Gartner advises that you don't build your own API management, but instead use an established API management platform to save time and reduce complexity.

Zazueta's first foray into API management was as the integration manager of a company that had a strong API program with 700 partners and 7,000 API consumers. "I was constantly on the phone and email troubleshooting developer issues, identifying problems with our API, going back to our engineering team negotiating ways to fix it," says Zazueta. "I became the de facto product manager for the API. It would have been so much easier if I had had the kind of stuff that you get with good API management."

The API Life Cycle

Full life cycle API management covers planning, design, development, testing, publication, operation, maintenance, and retirement of APIs. Your API management platform should support the entire life cycle:

Planning and designing the API
: Planning and design shouldn't just cover basic functionality, but also nonfunctional requirements like security and scalability and the definition of KPIs.

Developing the API
: APIs can be developed in a variety of different tools or frameworks. If you are using a microservices architecture, the implementation technology can change easily while preserving the API itself.

Testing the API
: After APIs have been implemented, you should put them through automated testing.

Deploying the API
: APIs should be integrated with your continuous integration environment and included in other DevOps processes so that they can be updated and redeployed quickly and automatically. The Swagger description format—now known as the Open API Initiative (OAI)—provides a standard format to describe the services supplied by RESTful APIs helping to automate configuration and speed deployment.

Operating the API
: The performance of your APIs should be monitored. Availability, performance, and response time standards must be maintained and all other KPIs tracked and analyzed. Enterprises might also define different levels of API access for different types of users that need to be enforced during operation.

Versioning and retiring the API
: You must have a plan to deal with and communicate different kinds of changes to developers. Nonbreaking changes are generally additions—creating new services or adding new parameters to existing services—which will not cause existing API integrations to stop working. Breaking changes remove services or parameters, change names or formats, and can therefore cause code calling your API to stop working. Older versions of the API might need to be maintained to give developers time to update their code.

Scalability, Security, and Support

Enterprise APIs should be secure, scalable, and available. If you create a popular or data-heavy API program, backend systems might struggle to keep up with demand. Some API platforms alleviate this problem with edge caching for data that does not change very often.

Traffic limits can be placed on individual developers or developer types, especially when API access is tiered and monetized.

Tight controls should be put in place to ensure that only approved developers have access to sensitive data accessed through APIs, while still making it easy for developers to gain access to the data they need. Adopting design-driven development practices for APIs—modeling the interface using a description language like the OAI spec before writing the supporting code—can help highlight potential data security issues early. In fact, using a contract-first approach in which the API is modeled and tested before being implemented is a recommended best practice.

Good API support is difficult and is often a neglected area in API programs. Providing a developer portal with service discovery, solid documentation, self-service API key registration, and onboarding is a minimum requirement. Good static and interactive documentation, tutorials describing how to implement common use cases, and support forums all help create a better developer experience. Support is just as crucial for internal API programs.

"You can't neglect a developer, internal or external, who is trying to get something done and just can't figure it out," says Zazueta. "You need folks who are technical enough to read another person's code and also charismatic enough to be a good support person. That kind of support structure is a really tricky thing to build."

Tools for Edge Integrators

Departments and lines of business increasingly want to create their own applications and integrations without the use of core IT development resources. Edge IT will focus on speed, agility, and experimentation to develop new functionality close to the lines of business.

However, without proper governance, self-service integration can cause security violations, data sprawl, and a brittle architecture caused by unmanageable point-to-point integrations. It is the responsibility of core IT to source edge integration tools and provide the governance needed to allow lines of business to produce enterprise-grade applications and work as independently as possible. This approach ensures that integrations comply with corporate policies and minimizes the number of tools used within the organization, while also offering lines of business the convenience they want.

Edge developers and integrators will not necessarily be IT specialists, and they certainly won't be integration specialists. Gartner defines a couple of different edge integration roles: ad hoc integrators are IT personnel like line-of-business developers who occasionally perform integration tasks. Citizen integrators are business users, from marketers to data scientists, who have a solid understanding of business processes, but often don't have development skills. They might want to create new applications, in which case they need integration capabilities, or just want to do something as simple as move some data from SaaS service to another.

Business users will need prebuilt templates and out-of-the-box wizards for developing simple integration flows. Line-of-business developers will need orchestration capability for developing complex integration flows. APIs are crucial for all these tools because they make it easier to create applications and integrations in low-code or no-code environments.

iPaaS for Line-of-Business Developers

At a minimum, line-of-business developers need the integration capabilities provided by API management and an iPaaS. An iPaaS lets users implement data, application, API and process integration projects spanning cloud and on-premises endpoints by developing, deploying, managing, and monitoring simple or complex integration flows between these endpoints. iPaaS capabilities typically include built-in connectors for various communication protocols, SaaS tools, and data sources and also support routing, orchestration, and integration flow life-cycle tools.

Because standard integration adaptors won't be suitable for all use cases, some iPaaS providers offer connector templates that you can customize. In most iPaaS products, developers must do this customization.

The level of coding skill or other technical knowledge required to use an iPaaS varies between tools. Some iPaaS offer no-code or low-code web interfaces that could potentially be used by citizen integrators, whereas others require an IDE and are therefore more likely to be restricted to developers. Model-driven iPaaS, which minimize coding, are some of the fastest-growing cloud platform services, representing more than $2 billion in annual revenue in 2016.

aPaaS for Citizen Developers

An aPaaS solution enables business users to rapidly build applications using visual models. aPaas also often have one-click deployment to public cloud, private cloud, or on-premises deployment environments. Gartner calls this genre of aPaaS *high-productivity aPaaS*, whereas Forrester refers to these tools as *low-code development platforms*.

"We personally think the space for people who want to build things in a visual manner without writing code is going to increase," says Rahul Kamdar, director of product management and strategy at TIBCO. "Technology is becoming an inherent part of everybody's job but not everybody is going to want to write code. aPaas allow people with reasonable amount of technology understanding but a good amount of business competency to start delivering applications and results in a more rapid manner."

Integration is an important capability of any aPaaS. In smaller organizations, simple database queries supply all the integration required to connect low-code apps to corporate systems. In larger enterprises, low-code applications might need to connect to a variety of legacy enterprise applications or data stores. Many aPaaS still require coding for at least some integration flows.

iSaaS for Citizen Integrators

iSaaS (integration software as a service) tools offer business users a very easy-to-use interface for performing simple integration tasks; no coding at all is required. iSaaS offerings allow business users to build services that are not mission critical for the enterprise but relevant to that specific business user or group within a line of business.

For example, a marketer might want start a drip email campaign in marketing automation SaaS tool Marketo for each new lead. Traditionally, this would mean writing code to integrate between the Salesforce and Marketo APIs. In an iSaaS, the marketer can use a web browser interface to define Salesforce as a source and Marketo as a destination and identify the addition of a new lead to Salesforce as a trigger event. When a new lead comes in, the new integration flow can move the lead from Salesforce to Marketo and associate it with the right drip campaign.

Microservices to Serverless

Microservices provide an organization with a set of reusable, lightweight services that can evolve independently to help enterprises rapidly react to change. Microservices are important in both core and edge IT. You can use microservices to develop core legacy systems and expose core data and services to the edge where they can be used by line-of-business developers and business users to rapidly create new applications and integrations in low-code or no-code environments.

Microservices are stateless, distributed, and independent and can be deployed to public, private, or hybrid clouds or edge devices and gateways. The hallmarks of a microservices environment are frequent releases, incremental service updates, and the need to monitor a constantly changing application topology. Developers and operations teams need to work together closely to build, deploy, and monitor microservices.

Challenges of Microservices

Microservices architectures pose their own challenges because they create a system with many moving parts. Best practices and microservices tooling help combat these challenges.

Having many small, independent services introduces complexity, especially across a large enterprise. When new features are needed, will a new microservice be created, or will an existing one be extended? Large systems often have services that overlap but are inconsistent in their naming and format. Defining a service contract, or terms of use, in a standard description format like Open API Initiative helps.

But each microservice can still be used in a variety of contexts. Will a microservice be able to scale in a particular context without affecting the quality of other services? How can security be maintained across many independent services? Applying scalability and security policies for different contexts can make it possible to reuse microservices instantly without having to modify the service itself. An

API gateway can enforce security, scalability, and other policies for each microservice.

Orchestrating the operation of a large number of microservices is another technical challenge introduced by the architecture itself. Choreographing the interaction between microservices and presenting them as a single composite service or API can make them easier to consume.

Creating many services that constantly interact creates many potential points of failure. Tracing performance problems is difficult when a single business transaction involves multiple microservices. Traditional logging is ineffective because microservices are stateless, distributed, and independent.

Microservices Tooling

Good microservices tooling can help organizations to meet the aforementioned challenges:

Service registration and discovery
: Microservices architectures are constantly evolving. A service registry contains information on how to reach each service and is updated when anything changes. Developers need to discover existing microservices and retrieve documentation not only on their technical interfaces, but on their service contracts, too.

Composition
: Because microservices are fine grained, clients often need to interact with multiple services. Microservices exposed via APIs can be composed using standard integration technologies. Netflix, for example, has hundreds of services and many different device types that it needs to serve. Intermediary services compose multiple other microservices in order to expose a simple custom interface to each device type. The Netflix API gateway provides each client with an API that's best suited to its requirements.

Configuration management
: Microservices might run in different datacenters, on a multitude of host machines or in containers. Each microservice can have its own local configuration file, but typically it is better to centralize management in a configuration server. Consul, for exam-

ple, can manage service discovery, routing, and configuration of microservices.

Circuit breaker patterns
When one service synchronously invokes another, if the invoked service is unavailable, this can lead to the calling service being unable to handle other requests. The failure of one service can cascade to other services. Instead of invoking synchronously, a service client should invoke a remote service via a proxy that functions in a similar fashion to an electrical circuit breaker. When the number of consecutive failures crosses a threshold, the circuit breaker trips, and for the duration of a timeout period all attempts to invoke the remote service will fail immediately. You can implement circuit breakers using tools like Hystrix.

Serverless Architectures

Serverless architectures are not actually serverless. Cloud computing originally virtualized hardware servers by using virtual machines (VMs). Containers virtualize the next layer up—the operating system (OS)—but don't completely hide the underlying infrastructure. Serverless computing, also known as function platform as a service (fPaaS), executes logic in environments with no visible VM or OS, thereby completely relieving developers of the need to manage the underlying infrastructure.

In serverless computing, very fine-grained units of application logic are packaged as functions or single operations and triggered by an event like an API call. Functions can be very cost effective because resources such as memory and CPU are allocated to the function only while it executes after being triggered. fPaaS providers, such as AWS Lambda, can charge based on how much memory and CPU resources are used with each invocation rather than per VM or container per hour.

fPaaS is only suitable for delivering a subset of application functionality. For example, performance on fPaaS can be variable, so serverless computing might not meet the latency requirements of mission-critical applications. In spite of this, Gartner predicts that by 2022, most PaaS offerings will evolve to a fundamentally serverless model.

Serverless can form part of a hybrid on-premises and cloud architecture. Events can trigger a set of processing tasks performed by a set of functions in sequence or asynchronously, and then the results are returned to the workflow to continue in the on-premises/virtual private cloud architecture.

Functions are inherently event driven because functions are invoked by a triggering event. Functions are also complementary to microservices because incoming requests from microservices are natural triggering events for functions. Most fPaaS providers allow functions to be triggered as a response to inbound requests, typically in some kind of API gateway.

API management becomes even more critical in a serverless world to manage the sprawl of microservices and functions arising from fine-grained application services. For example, if an enterprise is using multiple serverless providers, an event-driven API gateway can route to the relevant provider for logic execution. As with microservice platforms, new tooling is needed to manage function description, metadata, dependencies, monitoring, and debugging.

Event-Driven Architectures

Event-driven architectures focus on processing and reacting to events, often in real time. Highly decoupled, single-purpose event processing components like microservices or functions can asynchronously receive and process events. A service publishes events when something notable happens, and other services subscribe to those events and react on receiving them.

Event-driven systems enable businesses to react automatically to events in real time, and as such are an essential building block of a digital business. Event-driven systems also shift the emphasis from data at rest—big data sitting in a data lake—to data in motion such as streams of sensor data coming in from IoT devices or user click streams.

Machine learning and real-time analytics will play a prominent part in the event-driven systems of the enterprise. Machine learning lets businesses learn automatically, but to create true digital business agility, learned models must be compared to real-time events so that businesses can make decisions and act automatically based on the latest data. For example, a new customer activity event might need to be combined with the customer's purchase history over the previous six months to decide what type of real-time offer to make to the customer.

Gartner predicted that event-driven architectures would go mainstream in 2017. By 2020 the analyst firm expects that achieving competence in event-driven IT will be a top-three priority for the majority of global enterprise CIOs.

Event-Driven Technologies

Event-driven systems have long been used in niche use cases like financial trading systems, so technologies that support event-driven design, such as message queueing, publish-and-subscribe systems, and stream-processing middleware, have existed for a long time.

Microservices that communicate with one another using events are more independent, so they are easier to update and scale out independently. Serverless functions are inherently event driven because they are typically triggered by the arrival of a new event. Microservices, events, and serverless functions must all be integrated to achieve an efficient digital business.

However, most enterprise IT infrastructure supports only request-driven API interactions (a service calls an API and waits for a response) and almost entirely ignores event-driven operations. A digital business platform will rely on a combination of event-driven and request-driven systems seamlessly integrated to achieve the desired result. Event streams will need to be connected to microservices, apps, data, and all other enterprise resources.

"We were actually thinking about things like an event-enabled enterprise almost 10 years back," said Rahul Kamdar, director of product management and strategy at TIBCO. "We worked on a suite of products that do things like complex event processing, real-time rules, stuff like that. Now our goal is to take a lot of those capabilities but make them work in the current context."

Event-driven systems typically capture events from streaming data like a user's clickstream, gameplay metrics or sensor data from IoT devices. The incoming stream is often being examined to determine which events are of interest, given that not every event requires a reaction.

Apache Storm is an example of an open source project for stream processing. Services can publish and subscribe to events via a message queue like Apache Kafka or TIBCO's enterprise messaging product, FTL. Storm processes streaming data at scale; Kafka and FTL process messages at scale.

New events need to be placed in context before they can be acted upon. Incoming IoT sensor events, for example, might need to be combined with stateful data like the version and location of the device which generated the event, real-time analytics calculated on the sensor data stream, or machine learning models. In-memory databases like VoltDB or MemSQL allow quick access to stateful data.

Inference rules allow event-driven systems to make quick decisions; for example, "If event A comes in, and the following conditions are satisfied, perform the following actions." Those conditions might come from an incoming event, from stateful data, or a combination of the two. Event responses like policy decisions or alerts from IoT devices are put back on the message queue to be processed by the next service.

Tooling for Event-Driven Systems

There are considerable gaps in tooling for enterprise-grade, event-driven systems. There is no ubiquitous protocol for event-driven communications that's comparable to the request-driven REST. There are no standard event governance solutions. There's a lack of low- and no-code development tools for event-driven solutions.

Complex event-driven systems use a mediator, or a mediation layer, to determine the order of the steps required to process an event. For example, the arrival of a stock trade event might require a mediator to validate the trade, check the compliance of that trade against various rules, assign the trade to a broker, calculate the commission, and, finally, place the trade with that broker. Mediation solutions for event-driven systems should also include monitoring, logging, security, governance, discovery and addressing, routing, and so on. Publish-and-subscribe middleware like Kafka provides only very basic event mediation.

Event-Driven API Management

API gateways can provide intelligent mediation, but full life-cycle API management tools are typically designed to support only request–response APIs. You cannot use such tools to create event-driven APIs using technologies like webhooks, which are used to subscribe to events, server-sent events (SSEs), and HTML5 Web-Sockets, or translate incoming request–response API calls into event triggers.

API management must evolve to support both request- and event-driven APIs. You can harness the mediation capabilities of publish-and-subscribe middleware, API management, and iPaaS (which can often mediate both requests and events) to work together.

Edge Intelligence and the IoT

Gartner predicts that 20.4 billion IoT devices will be in the field by 2020. IoT device sensors will produce vast amounts of high-velocity, streaming data that must be analyzed and acted upon in real time. At the same time, IoT devices have limited computing power and unreliable connectivity, making it necessary to limit the amount of device data that is streamed back to the cloud for analysis.

To solve this problem, edge computing pushes intelligence, processing power, and communication capabilities directly into IoT gateways or IoT devices themselves. Manufacturing, utilities, and transport companies, for example, will monitor their hardware assets in order to respond to anomalies locally. Edge intelligence will become an important facet of an enterprise's digital business strategy.

To create edge intelligence, IoT services must be able to sense, connect, learn, and act as locally as possible in a constrained environment:

Sense
: Ingest events from sensors and other event sources.

Connect
: Collect, aggregate, and transform data from heterogeneous devices supporting numerous communication protocols, data types, and transport requirements, and connect back to cloud and on-premises apps.

Learn
: Use data from event streams in supervised, semi-supervised, and unsupervised learning.

Act
: Make decisions in real time with pretrained models or declarative logic and act on those decisions.

IoT services require an event-driven architecture and will use microservices and serverless functions. APIs can be used to securely expose connected devices, so APIs will proliferate as the number of IoT devices increases. API management for IoT must be event driven as well as request driven, have a microfootprint so that it can be deployed on IoT gateways, and deal with streaming real-time data and serverless functions. The IoT use case, therefore, brings together many of the technologies discussed in this report. All of those technologies must be integrated.

In a recent Gartner survey, 43% of respondents identified integration as one of their top three IoT challenges. "Through 2018, half the cost of implementing IoT solutions will be spent integrating various IoT components with each other and backend systems," says Benoit Lheureux, research vice president at Gartner. "It is vital to understand integration is a crucial IoT competency."

Event-Driven API Management with Mashling

APIs will proliferate as the number of IoT devices increase, but traditional request-driven API management products do not support essential IoT features such as event-driven APIs, serverless functions, and microservices choreography. Nor do they have the small footprint required to deploy API management on an IoT gateway.

Mashling is an open source project that implements event-driven, lightweight API management for IoT. Mashling is aimed at developers creating microservices to be deployed on edge gateways or devices. If an IoT gateway has services that are not fronted by APIs, a developer can build those APIs in Mashling, and deploy the Mashling engine to the IoT gateway to put those APIs in front of the edge services. A triggering event might arrive on a MQ Telemetry Transport (MQTT) queue or similar communications protocol.

Edge Intelligence with Flogo

Project Flogo is an open source, ultralightweight, edge microservices framework enabling application logic to be deployed to edge devices such as microcontrollers, or FaaS platforms such as AWS Lambda. Flogo was built from the ground up to reimagine integration for the event-driven age of computing, starting with IoT. The runtime footprint of Flogo's integration engine is up to 20 times lighter than Node.js and 50 times lighter than Java. Start time is in the millisecond range.

"I think edge intelligence is going to expand to just intelligence anywhere, so building intelligence in the Flogo engine enables us to provide analytical capabilities or predictive capabilities to cloud based applications, to serverless functions, behind the firewall and in edge devices," said Matt Ellis, product manager of Project Flogo.

To enable edge intelligence, incoming events must be enriched with contextual metadata stored in a stateful data store. Big data analytics like thresholds, profiles, and machine learning models might need to be combined with incoming sensor data and contextual metadata in order to make decisions. Alerts, alarms, and policy decisions must be exported to downstream systems and filtered IoT data sent back to big data systems in the cloud.

Flogo allows IoT gateways and devices to gather sensor data using IoT connectivity protocols like Bluetooth low energy (BLE), MQTT, and constrained application protocol (CoAP) and to perform logic like conditional checks or aggregation to make decisions. Gateways can also easily integrate with other apps and cloud services via messaging patterns such as MQTT, CoAP, WebSockets, or TIBCO eFTL.

Flogo Edge Apps moves the ability to perform conditional logic into IoT devices themselves, enabling them to autonomously make decisions and take actions. Developers can define this conditional logic, which Flogo calls a flow, as shown in Figure 1-1, in a no-code environment and deploy directly to devices.

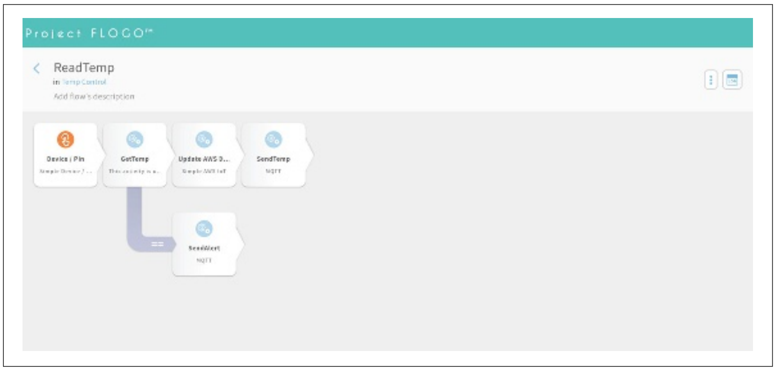

Figure 1-1. A Flogo flow

For example, as demonstrated in Figure 1-1, a flow might say that if the temperature of a connected sensor reaches a specific threshold, an action should be taken on the device. Flows can check stateful data via the Flogo state service and send state changes to local or cloud services using messaging technologies like MQTT or Kafka or TIBCO's eFTL. A flow can also be executed as a serverless function.

Machine learning models, encapsulated in a model format like TensorFlow's Protobuf or Predictive Model Markup Language (PMML) can be delivered to Flogo for inferencing inline, within a Flogo flow. The Flogo open source project will support a contribution model that allows new model formats to be added. Google TensorFlow is the first framework and model format that will be implemented in Flogo.

Models often use preprocessed data as an input. "We're introducing some streaming-like operators to Flogo, things like aggregate operators with sliding windows," says Ellis. Data from devices might also need be transferred to the cloud in order to be incorporated in model training or analysis. Flogo makes it easy to transfer raw or aggregate data, or the model output itself, to remote endpoints for future analysis and processing.

A Complete Pervasive Integration Solution

"The question for our largest, most long-standing enterprise customers is, how do I convert my mass into momentum?" says Rajeev Kozhikkattuthodi, vice president of product management at TIBCO Software.

Momentum is mass multiplied by velocity. A digital business platform helps enterprises to harness all of their mass—hardware, software, people, and processes—to create momentum by reacting quickly to change. An enterprise that doesn't just connect and sense, but also learns and acts automatically, can constantly evolve in an intelligent way.

A core requirement of a robust digital business platform is the ability to support pervasive integrtaion. TIBCO provides a complete pervasive integration solution serving every audience (business users, line-of-business developers, and core IT teams) and every integration use case in your organization. TIBCO's solution supports technologies like microservices, event-driven architectures, serverless and FaaS, edge computing, and machine learning.

Pervasive integration needs vary by audience (see Figure 1-2). Enterprise IT must provide developers and business users with secure access to data and services in the cloud, on-premise, or on edge devices, while enforcing corporate policies. Developers need to build and deploy new services at speed to on-premise servers, in the cloud, and out to IoT devices. Business users want to take advantage of their domain expertise to improve efficiency and digitize processes at the edge of the organization. TIBCO's pervasive integration solution serves all of these needs.

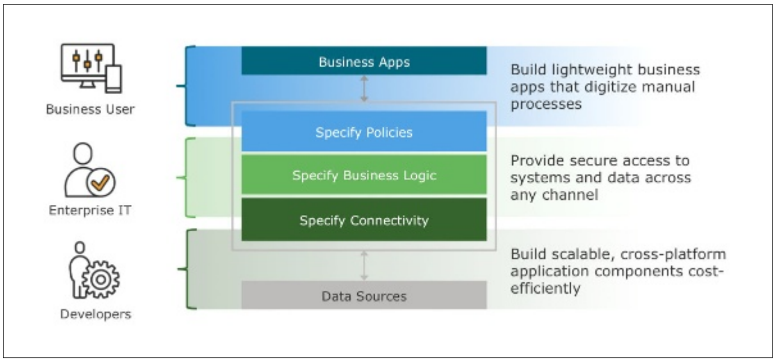
Business User
Enterprise IT
Developers
Business Apps
Specify Policies
Specify Business Logic
Specify Connectivity
Data Sources
Build lightweight business apps that digitize manual processes
Provide secure access to systems and data across any channel
Build scalable, cross-platform application components cost-efficiently

Figure 1-2. Pervasive integration needs by audience

Using TIBCO's pervasive integration solution (Figure 1-3), IT can safely expose services, data, and devices via APIs, microservices, and events. Developers can access core IT microservices, data, and devices via APIs, connectors, and message queues while using machine learning models, even on edge devices. Business users can use APIs and SaaS tools to digitize local processes, analyze data, and build their own simple services.

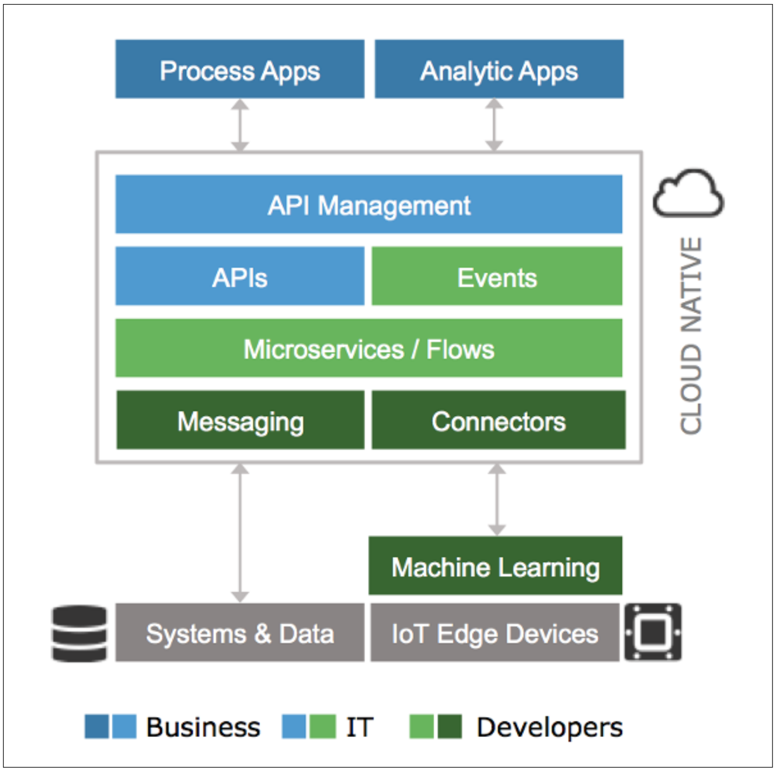

Figure 1-3. The modern digital stack

TIBCO's pervasive integration solution helps enterprises to build a digital technology platform for any industry, to sense and connect, and to learn and act. Turn your mass into momentum today, and your organization will be ready to transform tomorrow.

About the Author

Ciara Byrne started her career in academic machine learning research, was the CTO of a security startup, and managed a suite of software products, as well as building her own.

Her writing has appeared in *Fast Company*, Forbes, *MIT Technology Review*, VentureBeat, O'Reilly Radar, TechCrunch and *The New York Times* Digital.

In 2014, she was shortlisted for the Knight Science Journalism Fellowship At MIT and was selected as a Significance Labs Fellow to build apps for low-income Americans.