APPLICATION DEVELOPMENT HANDBOOK

FEBRUARY 2017

ISTOCK

Microservices development presents integration, deployment challenges

TechTarget

In this handbook:

Solving microservices modern integration issues

FRED CHURCHVILLE

In the development world, new advancements such as microservices, containers and APIs are somewhat of a double-edged sword. How will they fit in with existing architectures, often based on traditional concepts like SOA? How should they fit into a continuous integration strategy? And as APIs become essential to data integration, what management challenges will follow? These are just a few of the challenges that lurk in the world of modern integration.

Many businesses, for example, may be trying to determine how they can merge SOA with microservices. But since microservices evolved from the REST model, they represent a completely different architecture paradigm than SOA. Consultant Tom Nolle points out that these differences could cause significant issues, including processing delays. His solution rests in the establishment software methods and design patterns that accomplish clearly defined operational goals.

In this handbook:

Another issue is the process of enabling continuous integration via containers. One solution is the creation of logically interdependent container communities. It's then important to reference those communities when mapping container deployments, because integration component communities must stay as close in line with production deployment as possible.

And we can't forget about APIs. Journalist George Lawton says 2016 was the year of the B2B API. But enabling access to B2B API services is not a free ride. The greatest challenge will be cutting through market hype to determine which APIs and API management services can deliver real value.

This handbook will not only highlight the biggest challenges of modern integration, but also help you understand what steps need to be taken to successfully combine microservices with SOA.

In this handbook:

Managing the merger of SOA and microservices

TOM NOLLE

Most enterprises have some experience with SOA in developing and deploying applications, and most of those companies now have some experience with microservices -- another service-based component model that some say is competitive with SOA and others say is complementary. To align your own SOA and microservices initiatives, create a definition of the goals of both methodologies that suit your operation, delineate software methods that meet those goals, define your merger and set specific architectural rules and design patterns to enforce those methods.

An interesting paradox emerges when thinking of services as the basis for application development. While most planners believe the original SOA model is similar to the microservices model now emerging, much of the formal documentation still separates SOA and microservices. If we're to evolve SOA to support microservices, that formal separation has to be resolved first. To do that, you have to accept that both SOA and microservices are probably defined

In this handbook:

incorrectly most of the time, partly because an intermediary service model is often overlooked.

EXAMINING SOA AND REST

SOA was designed primarily to support the flexible reuse of components across applications, building on the principles of modularity in software design that emerged decades ago. With SOA, modules bound directly into application images were separated and coupled through a network connection as services. There were few restrictions on the function of these services, just as there are few restrictions on what a module can do. Instead, emphasis was on discovery through registries to allow developers to find useful services.

When the web became popular and web-enabled applications exploded, an alternative model emerged in the HTTP-based interactions used by web servers. Representational State Transfer (REST) models function not explicitly as a processing component but rather as a resource. RESTful design would represent an answer, while traditional SOA would represent a process. To use RESTful design, architects and developers would shift from using modules that

In this handbook:

were network-linked to getting answers that were location-independent. REST is therefore conceptually different from SOA.

That's critical in merging microservices and SOA because microservices evolve from the REST model, so they represent a different software architecture paradigm than SOA did initially. IBM defines a model where merging the two methodologies puts microservices at a lower level. Another model is where SOA is a layer below microservices. So which is right? It goes back to REST.

RESTful components -- if they're authored truly as resources -- would be stateless, sharable, and scalable or replaceable. Those properties are highly desirable if not essential in cloud-ready applications. So for cloud development, it would make sense to adopt the SOA-under-microservices approach.

A microservice is a small, generally useful piece of functionality that's accessed over the network via a RESTful interface. To put SOA under a microservices front, we'd have to implement it partly using SOA, which means harmonizing SOA with the RESTful front-end interface. How easy that would

In this handbook:

be depends on the complexity of the implementation and the specific SOA details involved.

In theory, it's possible to embed a multicomponent SOA implementation in a microservice, but the practice would have some significant risks. The first step in making it work would be to ensure that the basic REST properties aren't compromised, which means that the SOA components would have to manage any stateful behavior and be able to present an answer or resource-modeled interface. That's easier to do if a single SOA component is encapsulated in a microservice rather than a sequence of components.

State control issues can be difficult to resolve optimally, without making changes to the SOA implementation. Stateful components hold values between activations, making it difficult to scale, replace or share them. In some cases, it's possible to use a database to retain state on a given transaction. But that could introduce a processing delay when multiple instances of a service are deployed because the database could be local only to one of them. Therefore, use that approach only if you're sure it will work.

In this handbook:

IMPACT OF SOA AND MICROSERVICES

Merging SOA and microservices can also result in a "culture conflict" because SOA is normally used with discovery and registry processes that facilitate component binding and prevent mismatched parameters. As SOA has evolved, some of the tools used for registration and discovery have evolved to a broader mission, and it's smart to check all your SOA registry contenders to see if one will serve microservices applications, too.

With microservices, explicit coordination of binding is often not the case; more care will be necessary in the application architecture and development processes to implement procedures such as API registries to resolve the SOA and microservices discovery differences and ensure proper use. Cloud providers like Amazon and Microsoft provide API registry services, and there are tools available in both open source form (WS02 Governance Registry or the Fabric8 development platform) and proprietary form (SwaggerHub or Microsoft, IBM and Oracle as part of a development platform).

These tools illustrate an important point about microservices, made even more important where SOA integration is required. It's essential to adopt a standard

In this handbook:

framework for your APIs that's supported by design tools and patterns at the developer level. Since some development work will be necessary to frame SOA elements as microservices anyway, that will be a good time to put design patterns into place to enforce a consistent API model across microservices. This practice will pay dividends down the line because it will let developers integrate SOA in a standardized way.

The convergence of API and SOA registries will raise the long-term question of whether a commitment to either microservices or SOA is best, even though merging the two methodologies is possible. For the near term, you're likely to strike a microservices-side balance if you have a significant commitment to cloud development and an SOA-side balance if you're predominantly augmenting traditional data center computing with web technology. In the long term, as is the case with so many other computing-policy agenda items, it will depend on where the cloud goes.

In this handbook:

Continuous integration and deployment with containers

TOM NOLLE

Programming is a team activity, which means it's a combination of individual programming and cooperative specification and integration. Once programmers are assigned specific components to code, they work on their own tasks and eventually merge them into a broader test framework that culminates in a complete application test. This process takes time, and continuous integration seeks to shorten the cycle by assigning application components to permit early integration, building test platforms that use the best versions of all components and structuring the transition to ALM from these individual test steps. Container deployment can complicate the problems of continuous integration because development teams often don't test in a container environment, limiting the utility of their early integration work.

In this handbook:

LOGISTICS OF CONTINUOUS INTEGRATION AND DEPLOYMENT

The basic notion of continuous integration is that the proper test framework for a given software tool involves the components with which it directly interacts. Therefore, software scheduling and assignments of resources should group components into logically interdependent communities based on which ones interact regularly. As the software development proceeds within each community, the test framework is regularly updated with the latest stable version of each component to ensure that early testing includes component logic as well as integration.

Most logical communities of components are created by natural superior/subordinate relationships. Module A may invoke Modules B, C and D, making them a natural community. In these superior/subordinate relationships, try to assign the superior elements for completion first so their linking function is continuously validated.

In most cases, your component communities will be joined by a small number of bridging components that link into multiple communities, and they should be assigned as soon as testing on communities is reaching completion. As

In this handbook:

component community testing reaches maturity, these bridging components will join communities into larger integrated elements, a process that will continue until the entire application has been assembled.

That means continuous integration is likely to retain the notion of levels of integration -- the unit-test and systems-test concepts of legacy development practices -- but these integration levels are built and rebuilt as often as confidence levels in the functionality of components changes. The number of integration levels and how software components are promoted to appear in a component community for a given level will depend on the complexity of the code or the changes being made over time. The process of staging components into testbeds is fundamental to continuous integration and deployment, but not all implementations will obey the same rules.

Think of component staging as a promotion process. When a component has been validated in a minimal level of independent testing, it's promoted to a test platform that consists of its component community. When the component has completed testing at that level, it's again promoted to a test platform hosting a bridged set of communities. This process continues until the component has been tested in all the platforms up to the full application level.

A component is added to a test platform so it can be used in testing other components based on a similar set of promotion policies. Most continuous integration users will release a component to be part of an integrated test platform when it has completed testing there, but some users will require that the component also be tested in the next platform up the line. The goal is to ensure that a component isn't certified for testing until it's unlikely further testing will require that it be revised, which might in turn mean other components would have to be retested.

FACILITATING CONTAINER DEPLOYMENT

The notion of component communities is very helpful in planning container deployments because a community would normally map to a container cluster (a Swarm in Docker). Deployment of containers depends on effective use of DevOps tools and container clusters to facilitate efficient resource use, easy connection of applications and users and effective scaling under load or redeployment in the event of a failure. This mapping to container deployment is important because it facilitates a key step; integration component communities must be created, maintained and tested within as close an approximation of production deployment as possible.

In this handbook:

Production deployment requirements for containers involve testing the operational lifecycle steps, including deployment, parameterization, scaling, failover and decommissioning. Since these steps will require some accommodation in the development process and the automation of the operational responses through the use of DevOps, the lifecycle steps must be introduced in the continuous integration process as early as possible. Otherwise, you risk incomplete testing and integration as well as defeating the fast-business-response goals of continuous integration.

A critical step in continuous integration is to establish that test framework, and just as critical is recognizing that the framework should be the same as that used for traditional application lifecycle management (ALM). Therefore, it's best to assume that continuous integration and deployment of container-based applications uses a set of container clusters, all identical and compatible with those in which the actual applications will be deployed. This testing-cluster model then seamlessly evolves to support ALM.

Container technology, especially when combined with features that allow for the generation of "portable clusters" that include a predefined set of components and applications, facilitates continuous integration. It reduces the

In this handbook:

penalty for sustaining test and production environments that represent the phases of testing from unit-code testing to system testing and deployment.

While any virtualization or cloud technology will introduce new integration issues, containers may well offer enhancements to the integration testing process that will more than make up for these issues. Containers go a long way toward the dream of having coding, testing and development practices live within an organized set of sandboxes along with tools to support each stage and practices to control progression of code to the ultimate goal of business use. With care, containers can promote rapid deployment goals overall.

In this handbook:

In 2016, B2B integration API strategy becomes a priority

GEORGE LAWTON

Enterprises are turning to an API strategy to drive B2B integration. Read on for tips to address the numerous challenges related to supporting legacy protocols, streamlining business processes and leveraging third-party data.

API gateways promise to streamline B2B integration, but enterprise architects need to address a number of challenges to make things work smoothly.

Both emerging and established businesses are enabling partnerships by offering access to their B2B API services. "For developers and enterprise architects, this means opening up internal business processes for external consumption, and the creation of middleware to connect consumers with providers," said Owen Garrett, head of products at NGINX Inc., based in San Francisco.

Security concerns will also mean enterprise architects need to begin to compartmentalize monolithic applications and business processes to isolate the

In this handbook:

hardened, secured external-facing interfaces from the sensitive internal systems. Monitoring and reporting is critical to the business -- both to measure compliance with service-level agreements, and to provide input to billing and metering processes.

MOVE TO UTILITY COMPUTE MODEL WITH APIS

"The most interesting trend around B2B modernization and integration relates to how IT shops are transforming to infrastructure that supports a utility compute model," said Mark Lewis, CEO at Formation Data Systems, based in Fremont, Calif. This allows organizations internal and external access, and provision resources via open, REST-based APIs, which enable the initiation of application compute and storage resources to be deployed on demand in a fully orchestrated, automated fashion.

One challenge in 2016 is how to transform manual processes into API-automated workflows, Lewis said. Another is how to migrate applications off of rigid, monolithic infrastructure to dynamic software-defined infrastructure in a secure manner, with the proper data governance controls in place. The most interesting development in 2015 around B2B integration was how the public

cloud platforms embraced B2B integration and have built a broad set of services to support this capability.

LEGACY INTEGRATIONS STILL IMPORTANT

Having a B2B integration API strategy is increasing and slowly overtaking more traditional integration methods. However, traditional protocols, such as AS2 or AS3 file-based integration, are still alive and will not disappear anytime soon.

"This calls for software solutions [that] can support the legacy integration models, while proposing new ways of integrating, such as APIs, event-driven integration or via messaging systems. Customers can then slowly switch over their partners to new integration channels," said Isabelle Mauny, vice president of products at WSO2.

APIS JUST THE BEGINNING

The simplification of integrations is shifting the business conversation. "Using technology, like APIs or webhooks, makes integrating two disparate systems significantly easier, but developers still have to program and manage those

In this handbook:

services, which, for enterprises, can be a big undertaking," said Neil Mansilla, vice president of developer relations at Runscope Inc., based in San Francisco.

Companies such as Zapier and IFTTT are proving themselves as useful players in the space as middlemen between organizations and the third parties with which they want to integrate. Mansilla said these services are transforming the integration space, allowing companies who can't invest in the technology, but need to partner with other organizations, do so easily and affordably.

Additionally, many newer companies are building their business as platform-first, creating hubs that put integrations with sometimes hundreds of services as a key offering of their product. Users can add integrations they need with the services they use in just a few clicks.

"As B2B integrations become more popular and vital to business, more services will crop up that facilitate complex integrations, and more companies will build out integration platforms to lower the barrier to entry for partnering," Mansilla said.

In this handbook:

HEALTHCARE MOVES TOWARD API INTEGRATION

Improvements in security and privacy infrastructure could also help to bring more B2B integration in the healthcare industry. WSO2's Mauny has seen increased demand for AS4 and FIHR -- in the healthcare industry -- and said she expects this to continue in 2016.

Today, many professionals in this space are still using Excel or fax to share data and information. The integration of carrier and provider data is already enabling a completely new set of services to address the $375 billion the insurance industry spends on paper-based data exchanges.

"As we begin to really feel the benefit of this integration in 2016, we'll see API-powered data integration become the norm instead of the exception in this industry," said Jason Andrew, CEO at Limelight Health.

THINK OF THE NETWORK AS AN INTEGRAL PART OF B2B STRATEGY

Businesses are increasingly rolling out cloud-based or Web-based portals to enable effective communication with their customers, suppliers and partners. The rationale behind putting a Web front end on these applications is that end

users can easily access them from any browser or from multiple devices. As user experience is one of the key performance indicators determining the success of B2B integration, it is crucial that these Web portals and applications do not suffer from sluggish performance.

The network should become an integral part of an organization's larger B2B integration strategy. "Enterprise architects should evaluate [content delivery network and software as a service] acceleration solutions to ensure application performance, and factor them into the strategy," said Gary Sevounts, chief marketing officer at Aryaka. If customers, suppliers and partners are globally distributed, a global private network with built-in optimization is crucial for success. A slow network can cause B2B integration initiatives to fail, especially if customers, suppliers and partners are based in remote locations, such as China.

EMBRACE NEW OPPORTUNITIES FOR OLD B2B INTEGRATION STANDARDS

Traditionally, B2B exchanges are expensive for customers to maintain. "With 2016 bringing an explosion of APIs, we'll see new opportunities for the old set of B2B integration standards," said Ken Yagen, vice president of products at

In this handbook:

MuleSoft Inc., based in San Francisco. API-led connectivity will allow organizations to tie together new software as a service and mobile applications with traditional B2B protocols and standards required to connect to partners. Yagen said he also expects more initiatives to appear, like the Directive on Payment Services that mandates standardized API access for payment services across the EU.

Companies will now be able to build reusable services across multiple B2B trading partners and B2B processes on a single connectivity platform. "This increases agility and decreases time for partner onboarding, while reducing cost and risk. APIs provide the gateway and visibility to integrate B2B exchanges that allows organizations to operate in a more efficient manner," Yagen said.

INTEGRATE THIRD-PARTY DATA INTO BUSINESS PROCESSES

In 2016, organizations will use more and more third-party data to drive business processes and decision making. However, integrating these data new sources across an organization's partner ecosystem is difficult, and puts pressure on legacy data management systems and processes. "The

In this handbook:

structure of these sources is beyond an organization's control and can change unexpectedly. These changes, called data drift, can corrupt data and break data operations. How organizations modernize to deal with data drift will be an important theme in 2016," said Arvind Prabhakar, CTO, StreamSets, based in San Francisco.

Cross-business flows of unstructured and semi-structured data are proliferating. This proliferation creates real problems for organizations. For instance, data engineers and scientists spend too much time cobbling data flow services together instead of pushing their organizations' data practices forward.

"Organizations that free up their data engineers and scientists to focus on furthering data practices will be able to cost-effectively create and maintain reliable and high-quality flows that support critical business decisions," Prabhakar said.

API GATEWAYS BECOME THE NEW STOREFRONT

B2B integration is rapidly evolving to digital business, as legacy integration methods, such as EDI, ESP, and ETL, are being replaced by a new breed of public API gateways.

In this handbook:

"Just as every company today has a website, every company will have a B2B API gateway,"said Ilan Sehayek, CTO at Jitterbit. "The companies that develop an integrated B2B API strategy ... will become the industry disrupters in their markets."

The greatest challenge will be cutting through the hype of the API marketplace to deliver real value to their partners. Too many B2B API projects require custom integration to achieve measurable return on investment, and APIs alone are not enough. According to Sehayek, business will need to implement a "hybrid" B2B API strategy that includes both existing and developing services.

"APIs will continue to evolve in 2016 to connect even more of the enterprise," Sehayek said. "Companies should be prepared to build out their API strategy with a greater focus on the end-to-end business process being exposed and according to Agile development principles."