E-book

Red Hat

Agile integration

The blueprint for enterprise architecture

redhat.com

Table of contents

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

"Modernizing IT is one particularly critical, and oftentimes misunderstood, part of digital transformation because it enables enterprises to accelerate innovation and performance improvement. Other essential transformation tasks include building a digital culture."[1]

Michael Bender and Paul Wilmot
Digital McKinsey

The role of integration in innovation

Business success is increasingly based on a company's ability to react to change. As new disruptive players enter markets and technology upends what consumers expect, organizations need to evolve to address these changes in much shorter cycles than ever before. Modern software architectures and processes can make organizations more effective at dealing with change and emerging as winners in their markets.

Entire industries have been transformed by technology. Today, most businesses offer e-commerce services, and consumers interact regularly with companies in the digital realm. This trend of disruptive change is leading organizations to radically transform their IT environments so they can deliver the new digital services that customers demand—better and faster than their competitors.

A new way of connecting applications and services across the enterprise, agile integration, is a fundamental solution. Agile integration combines three powerful architectural capabilities—distributed integration, application programming interfaces (APIs), and containers—to foster agility, power new processes, and ultimately deliver a competitive advantage.

Industries like travel and hospitality have been transformed by new ways of doing business—new services are now offered, and consumers interact with services differently. This trend of disruptive change is extending across other major industries—from financial services to government, spurred by new technologies and mindsets about how businesses and customers interact. These challenges are pushing existing organizations to radically transform their own IT technologies to deliver these new services.

Integration is essential for digital business

Delivering excellent customer experiences is no longer a matter of differentiation—it is a matter of survival. While individual experiences and interactions are the building blocks of customer loyalty, a holistic customer journey is the mortar that binds those blocks together.

At a hotel, for example, the stay is as important as the booking interaction, the conversation the guest had about Wi-Fi issues, the loyalty program, and so on. Customers are more demanding in the digital economy, and their expectations for personalized and context-relevant interactions have never been higher. Ultimately, the customer relationship is as strong as the weakest link on the customer journey.

To meet customer expectations today, an organization must facilitate seamless cross-app data sharing by integrating the many different applications the customer interacts with throughout their journey. A successful integration strategy will allow a company to generate multidimensional customer insights, anticipate customer needs, and minimize churn.

Uber is the perfect example. While many cite Uber as an example of digital disruption, the impact of integration on its success is often overlooked. Taxi companies have had access to the same customer data for at least 20 years, but they were previously unable to connect the applications and utilize the data in a way that created predictability and changed customer expectations. Agile integration bridges that gap, opening up almost unlimited opportunities for innovation in today's digital world, and transforming the way companies compete.

---

1 *Michael Bender and Paul Wilmott, Digital McKinsey, "Digital reinvention: Unlocking the 'how.'" January 2018,* https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/Digital%20Reinvention%20Unlocking%20the%20how/Digital-Reinvention_Unlocking-the-how.ashx

"If you can't out-experiment and beat your competitors in time to market and agility, you are sunk. Features are always a gamble. If you're lucky, ten percent will get the desired benefits. So the faster you can get those features to market and test them, the better off you'll be. Incidentally, you also pay back the business faster for the use of capital, which means the business starts making money faster, too."[2]

Gene Kim
The Phoenix Project

Today's most progressive market leaders are connecting everything, because they know that integration is vital to their digital transformation—and to their ultimate success.

Agile integration: Attaining flexibility and speed

Speed is critical in the digital world. To stay relevant, organizations must plan and execute changes to their software systems quickly. An organization that can change pricing or make new products available overnight has an enormous advantage over one that requires a three-month staged roll out with a cascade of manual verification steps.

To deliver software at the speed required by the digital economy, organizations need an agile infrastructure foundation. In this case, agile does not refer to agile software development. It refers to the more traditional meaning of agile—flexible, adaptable, and able to move quickly.[3]

To date, agile methodologies have focused on software development, with the goal of improving and streamlining how applications are built. DevOps[4] practices have tried to inject that methodology into how applications are deployed as well. DevOps only reaches so far, however, primarily addressing only new software applications developed by the organization.

Infrastructure agility goes much further, creating an environment that encompasses all IT systems, including legacy software. An agile infrastructure takes the complexity of existing systems, different data types, data streams, and customer expectations, and unifies them.

Red Hat calls this process "agile integration." Integration is not a subset of infrastructure—it is a conceptual approach to infrastructure that includes data and applications with hardware and platforms. By aligning integration technologies with agile and DevOps technologies, an organization can create a platform that provides development teams with the ability to change as quickly as the market demands.

Planning is dead: Organizations and agility

As Jim Whitehurst, Red Hat CEO, once said, "Planning as we know it is dead. … Planning in a less-known environment is ineffective."[5] As business environments speed up and change becomes more jarring, plans break quickly and being locked into one course of action can be extremely costly.

What that means is that the less information a company has, or the less stable the environment is, the less valuable plans are.

You don't know what you don't know

Infrastructure planning typically takes a long-term approach, sometimes spanning years. However, a multiyear plan can impede the ability to innovate or pivot as the market changes. Agility is the ability to plan more quickly and execute on those plans quickly. In this environment, plans have a shorter life expectancy and new plans are continuously cultivated.

---

2 *Gene Kim, Kevin Behr, and George Spafford, The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win. Portland, Oregon: IT Revolution Press, 2013.*

3 *Oxford English Dictionary*

4 *Understanding DevOps,* https://www.redhat.com/en/topics/devops

5 *Jim Whitehurst's keynote address at Red Hat Summit 2017.* https://www.youtube.com/watch?v=8MCbJmZQM9c

This rapid change can be challenging when teams are accustomed to six-month or even 24-month development cycles. The problem is exacerbated when more traditionally structured organizations must compete with startups that are approaching the market in entirely new ways. There are obvious examples with Netflix and Blockbuster, or Uber and traditional taxi services, but the disruptive effect of startups goes back to the earliest days of the Information Age, starting with Amazon in 1993 or personal computers in the 1980s.

Table 1. Disruptors across industries

Industry	Traditional service	Disruptor	Effects
Transportation	Taxis, public transit	Uber, Lyft	Creating uniform customer experience that is nearly impossible for small, local firms to replicate
Wealth management	Investment firms	Automated funds	Shifting fund management differentiators from personnel to algorithms
Retail	Physical shopping	Amazon	Changing shopping habits from offline to online purchasing
Search engines	Google, browser-based search	Voice search	Affecting Google's primary channel to market and allowing in new entrants

The advantage that startups and disruptors have is the flexibility to structure their infrastructure, teams, application, architecture, and even their deployment processes in new ways. It is more than just having an innovative idea. They are able to execute those ideas because they aren't held back by legacy infrastructure – or as Rachel Laycock jokingly put it, "legacy people."[6] They can be agile.

Beyond the ability to build something new, these organizations also build systems that are designed from the ground up to accommodate change. Software infrastructure is part of their differentiating power, and almost any part of the system can be swapped out, updated, or removed in response to changing market needs. As startups age, some suffer from a reduced ability to adapt, but the best organizations ensure their ability to change is protected at all costs.

6 *Rachel Laycock, "Continuous Delivery." Afternoon general session, Red Hat Summit–DevNation 2016. July 1, 2016, San Francisco, California. https://youtube.com/watch?v=y87SUSOfgTY*

Laying the foundation for agility

To succeed in fast-moving environments, the entire IT infrastructure must function in an agile manner.

Change needs to occur at two levels:

1. Organizational and cultural support of agile processes—from architectural design to team communication.
2. Technical infrastructure that allows users to upgrade, add, and remove capabilities rapidly.

Technical and cultural change do not create agility. They are the foundation for it.

Marty Cagan, product manager from eBay, applies what he calls a tax to every project—some time and resources are set aside from every routine project to work on new infrastructure projects.[7] This approach makes new projects and innovations a priority.

The infrastructure of agility

A barrage of new technology often does not help create an agile infrastructure since different groups move in different directions to explore options for improvement. Without a coherent set of top-level goals, it can be difficult to determine which set of new capabilities will make a genuine difference to the overall functioning of the organization.

The three pillars of agile integration

Three main technologies underpin an agile integration approach:

1. **Distributed integration:** A few dozen high-level integration patterns reflect enterprise work and dataflows. When these integration patterns are deployed within containers, they can be deployed at the scale and location needed for specific applications and teams. This approach represents a distributed integration architecture, rather than the traditional centralized integration architecture, and it allows individual teams to define and deploy the integration patterns that they need with agility.
2. **APIs:** Stable, well-managed APIs have a huge effect on collaboration between teams, development, and operations. APIs wrap key assets in stable, reusable interfaces, which can be used and reused as building blocks across the organization, with partners, and with customers. APIs can be deployed together with containers to different environments, allowing different users to interact with different sets of APIs.
3. **Containers:** For both API and distributed integration technologies, containers work as the underlying deployment platform. Containers allow the exact service to be deployed within a specific environment in a way that is easy and consistent to develop, test, and maintain. Because containers are the dominant platform for DevOps environments and microservices, using containers as the integration platform produces a much more transparent and collaborative relationship between development and infrastructure teams.

7 *Cagan, Marty, Inspired: How to Create Products Customers Love. Wiley Press, 2017.*

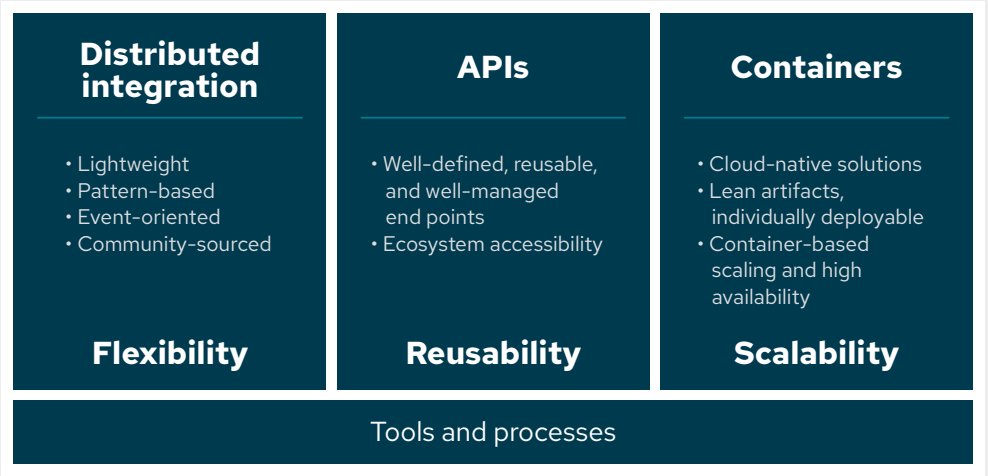

Figure 1. Three pillars of agile integration

The three pillars of agile integration make IT infrastructure more agile because they each raise the level of abstraction at which different teams can work together. Using a container platform with APIs and distributed integrations abstracts the implementation of the integration from the integration itself. Teams can be more agile because APIs and distributed integration patterns package specific assets at a level that can be broadly understood—without having to understand or alter the underlying infrastructure.

Individually, each of these technologies will provide significant agility to specific integration challenges. When used together, they provide a multiplier effect. Underscoring the technology is culture: The benefits of the technology are increased when combined with DevOps practices—especially automation and deployment processes.

Pillar 1: Distributed integration

One of the greatest challenges of traditional IT systems is the need to connect applications from across the organization and beyond. In the past, this need led to increasingly complex, centralized integration hubs. These hubs, often implemented as enterprise service buses (ESBs), have become extremely complex bottlenecks that are too rigid to allow rapid changes.

Using an ESB requires that the ESB's tools are used for the entire life cycle, in addition to whatever tools you use in the development and operations environments. This limitation leads to awkward, inefficient, and error-prone operations.

Distributed integration achieves many of the same technical objectives of previous generations of ESBs, but in a way that is more adaptive to teams within an organization. As with ESBs, distributed integration technology offers transformation, routing, parsing, error handling, and alerting capabilities.

"In software, when something is painful, the way to reduce the pain is to do it more frequently, not less."[8]

David Farley
Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation

The difference is the architecture of the integration. A distributed integration architecture treats each integration point as a separate and unique deployment, rather than part of a larger, centralized integration application. The integration can then be containerized and deployed locally for a specific project or team without impacting any other integrations deployed throughout the organization. This approach essentially treats integration as a microservice,[9] which increases the speed of development and supports rapid release cycles.

This distributed approach provides maximum flexibility. It also uses the same toolchain as the agile or DevOps teams by using the underlying container platform, enabling teams to manage their own integrations with their own tools and schedules.

Alignment with developer tools and processes is critical. Distributed integration is not a centralized software infrastructure developed and managed by a specialized set of users in one department and deployed separately from the software development process. Distributing the integration architecture, with a common platform and tooling, keeps it accessible to all developers at a project level and supports lightweight deployments wherever and whenever integration is needed.

Table 2. A comparison of integration technologies for each stage of the software life cycle

Life-cycle step	ESBs, most integration platform-as-a-service (iPaaS)	Supporting distributed integration technologies
Version control	Proprietary	GitHub and others
Build	Proprietary	Maven and others
Deploy	Proprietary	Containers and other DevOps tools
Manage and scale	Proprietary	Containers and other DevOps tools

Streaming over a distributed backplane

Agile integration offers the freedom to use either synchronous or asynchronous integration, depending on the needs of the application. A common distributed integration approach uses the synchronous method, deploying containers to share data between different users, as discussed above. A second distributed integration option is streaming, an asynchronous method, which allows agile developers to receive, reread and replay events from other sources as needed. Messaging is used to replicate data in an intermediate store, so the data can be shared between multiple teams and their applications. The intermediate datastore is advantageous to microservices teams, because they are not forced to continuously seek out data synchronously from other sources.

8 *David Farley and Jez Humble, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison-Wesley Professional, 2010.*

9 *See Martin Fowler's helpful definition of microservices:* https://martinfowler.com/articles/microservices.html

A distributed streaming platform can publish and subscribe to, store, and process streams of records, in real time. It is designed to handle data streams from multiple sources and deliver to multiple consumers. Streaming platforms can handle up to millions of data points per second, which can be especially beneficial in use cases that require real-time data availability, such as IT operations and ecommerce. The figure below illustrates a pattern representing many streaming use cases. All applications produce changes as a consequence of operation. These applications can propagate these changes as events to one or more streaming processors. In turn, streaming processors perform pattern matching or transformation, and then match the current situation with historical insights often provided by their analytics systems.

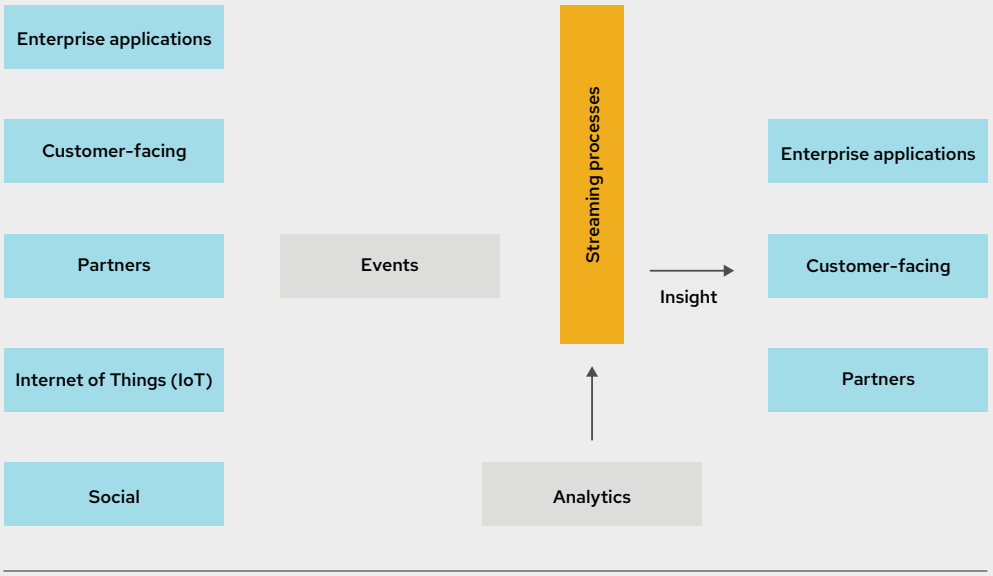

Figure 2. Events flow through processing platform to yield insight

Asynchronous integration provides advantages over synchronous integration, in some use cases. Synchronous integration requires that the receiver is available for communication to be successful, while messaging-based asynchronous integration allows a message to be sent even if the receiver is not available at the time. In synchronous communication, timeouts are common because the application must wait for a response in order to continue operating. This delay does not happen with asynchronous messaging, which continues processing regardless of whether a response was received. This flexibility makes asynchronous integration, such as streaming, ideal when handling large volumes of data. Asynchronous integration also delivers high availability and reliability because the system is less likely to time out.

Use of asynchronous event-driven integration, such as streaming, to augment synchronous integration and APIs, further strengthens agile integration. The combination of integration and messaging improves the overall performance of the integration architecture by offering more effective routing, support for multiple languages and protocols, high throughput, and superior data management.

Event mesh

Another new, asynchronous, distributed integration option is an event mesh, a configurable and dynamic infrastructure layer for distributing events among decoupled applications, devices, and clouds.

An event mesh, which is made up of a network of interconnected event brokers, handles asynchronous, event-based interactions. It is environment-agnostic, so events from one application can be routed and received by any other application no matter where they are deployed—including on-premise, private cloud, public cloud, hybrid cloud, Platform-as-a-Service (PaaS) or even the Internet of Things (IoT)—without configuring event routing.

In addition, an event mesh can connect everything from microservices and cloud-native services to legacy applications, devices, and databases—basically connecting any event producer to any event consumer. An event mesh is highly efficient, determining the fastest path between event producer and consumer, to enable real-time interactions.

As IT environments become increasingly distributed, event mesh provides a modern solution, enabling event communications to be flexible, reliable, fast, and more secure.

Asynchronous interactions and event-driven architectural patterns are not new, but event mesh is a ground-breaking new approach that is only starting to emerge in the integration arena. While event mesh is not widely adopted today, Red Hat expects this technology to gain traction in the coming years, as digital transformation requires organizations to become more reliant on the flexibility of event-based integration.

An event mesh is not the same as a service mesh. While event mesh is asynchronous, service mesh supports synchronous integration based on APIs.

Architecturally, distributed integration treats integrations as microservices. The integrations have the ability to be containerized, are easily and locally deployable, and can have rapid release cycles.

Integration technology must support a lightweight, microservices-based architecture. Red Hat® Fuse allows users to treat integrations as code, which can run anywhere—including in a container.

Additionally, Fuse is bundled with Red Hat AMQ to provide a strong messaging infrastructure, ensuring events and data are routed between systems effectively. Messaging is helpful when working with microservices because its asynchronous nature requires no dependencies.

Red Hat AMQ offers Apache Kafka—a distributed streaming platform—via AMQ streams on Red Hat OpenShift® Container Platform, the enterprise Kubernetes platform trusted by nearly half the Fortune 100.[10] AMQ streams is a massively scalable, distributed, high-performance data streaming capability based on the Apache Kafka project. This combination allows microservices and other applications to share data with extremely high throughput and low latency.

Red Hat also delivers an event mesh via AMQ interconnect, a global, peer-to-peer event delivery system. AMQ interconnect serves as a distributed traffic manager that routes around bottlenecks and failures to reach consumers simply and reliably. In this way, it offers resiliency to node and cloud failures.

Pillar 2: Application programming interfaces (APIs)

Most information infrastructures contain hundreds or even thousands of systems, applications, and assets, and it can be very difficult for these systems to interact—and it may not be possible for IT administrators to know what systems are even available. APIs solve this challenge by serving as the interfaces for anything that can be connected using integration technology.

As organizations shift from a centralized integration approach to a distributed approach, self-service becomes a key priority. Agile teams need the authority and autonomy to seek out, test, and use services developed both inside and outside of their companies. A strong API capability delivers this authority and autonomy to those teams. With APIs, the teams gain the integration flexibility they need while the organization can ensure that security, authorization, and usage policies are managed and enforced.

APIs deliver a set of definitions or rules that set up how applications communicate with each other, providing developers with a common language for integration and a reference on how to design integrations.

10 *Red Hat press release, "More than 1,000 enterprises across the globe adopt Red Hat OpenShift Container Platform to power business applications." May 8, 2019,* https://www.redhat.com/en/about/press-releases/more-1000-enterprises-across-globe-adopt-red-hat-openshift-container-platform-power-business-applications?extIdCarryOver=true&sc_cid=701f2000001OH74AAG

Developers use APIs as building blocks within their projects. But APIs are also meant to be shared. Different APIs, or different subsets of an API, can be made available to different audiences. For example, the needs of a vendor may be different from those of internal development teams or community developers, and the appropriate APIs can be exposed to each of these groups as needed.

To utilize APIs successfully, the organization must have API management capabilities. API management includes designing the API for the application and the user group, as well as managing the life cycle of the API. APIs are increasingly managed as products, with different teams responsible for each API, but there is a need to ensure uniformity and ease of use across all of these resources.

Service mesh

A service mesh takes API integration one step further by providing a configurable infrastructure layer for microservices, making communication flexible, reliable, fast, and manageable.

When an organization first uses microservices, built-in governance handles service-to-service communication with minimal disruption to operations. However, as new applications, services, and features are continuously added, in the form of microservices, at some point the operational complexity of the architecture can cause problems. Every new service introduces a new potential point of failure. Services can get overloaded with requests. Hundreds or thousands of microservices all constantly trying to connect to each other can result in communication latency and application downtime. In addition, it becomes almost impossible to identify the source of the problem in a complex microservices architecture. The service mesh was introduced to address these problems.

Service mesh is a dedicated infrastructure layer built right into an app. A service mesh removes the logic governing service-to-service communication from individual services and abstracts it to a layer of infrastructure. A sidecar proxy sits alongside a microservice and routes requests to other proxies. Together, these sidecars form a mesh network. In this way, the service mesh routes requests from one service to another, optimizing how all the microservices work together.

Service mesh allows users to introduce added capabilities to microservices, such as routing, fault tolerance, and security, as well as visibility, monitoring, and testing, while making no changes to the microservice components themselves. This capability is accomplished by the sidecar proxy injecting information and functionality into the microservice.

Service mesh makes it easier to troubleshoot communication failures because the source of the problem is not hidden within the microservice. Instead, it is found in a visible infrastructure layer alongside the services.

In addition, service mesh makes applications more robust and less vulnerable to downtime because a service mesh can reroute requests away from failed services.

Service mesh also collects performance metrics to provide greater diagnostic capabilities in the event of a service failure, significantly reducing downtime. This same performance data can also be used by developers to optimize designs in future releases.

Without a service mesh, each microservice must be coded with logic to enable service-to-service communication, which takes up developer time and focus. Because a service mesh eliminates the need for this extra coding, it streamlines the development process. As a result, Red Hat expects this technology to grow rapidly in popularity over the next few years.

The true power of APIs comes from allowing others to use them – both internal developers and external users. Red Hat 3scale API Management offers tools to help all these users. It provides a developer portal to foster collaboration on API creation, and an admin portal to publish the APIs. 3scale API Management also helps make APIs consumable externally by providing authentication, integrating with major cloud providers, and running inside containers.

API strategy combines API design with a way to take that API public. 3scale API Management, especially 3scale on top of a container platform, provides the means to execute that strategy.

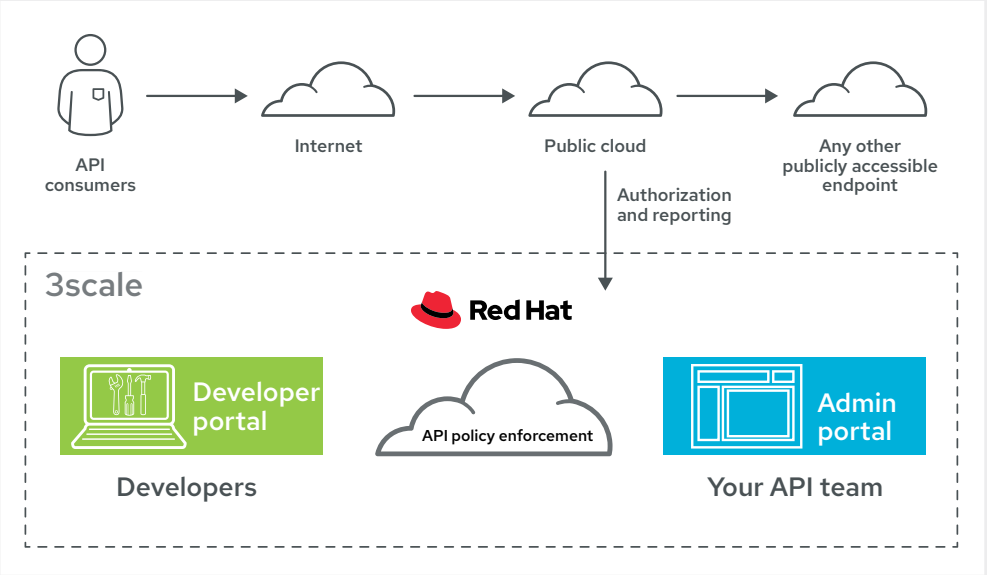

Figure 3. A view of API management, endpoints, and the public cloud

Red Hat also offers Red Hat OpenShift Service Mesh, which takes the popular Istio service mesh and combines it with other key projects, like Jaeger for tracing and Kiali for visualization, to provide better manageability and traceability to microservices deployments.

In addition, 3scale provides full API management capabilities to the Istio Service Mesh in Kubernetes using the 3scale Istio adapter.

"Getting multicloud right involves lots of thinking about processes, staff skills and organizational structures, but the industry agrees that the tech platforms that make multicloud feasible are containers and Kubernetes."[11]

Suresh Vasudevan
Forbes

Pillar 3: Containers

Virtualization, cloud, and containers technologies accomplish a similar goal: they abstract the software operating environment away from the physical hardware, making it possible to stack more instances on hardware and manage utilization, scale, and deployment more efficiently. However, they address that challenge in different ways. Virtualization abstracts the operating system layer. Cloud removes the concept of permanent, dedicated server instances. Containers define a just-enough version of an operating environment and libraries to run a single application.

The more prescriptive and lightweight approach provided by container technology makes it an ideal tool for modern software environments. Each instance uses an immutable definition, from the operating system to the exact version of each library included. The resulting unit is highly repeatable and consistent for each instance, which is well-suited to continuous integration and continuous delivery (CI/CD) pipelines. The combination of lightweight and repeatable makes containers the perfect technology platform for agile integration.

Additionally, because a container image only defines what is needed for a single unit of functionality, containers fulfill the vision of microservices, and container orchestration facilitates the deployment and management of large microservices infrastructures.

Traditional integration approaches had a highly centralized structure, with ESBs located at major points in the infrastructure. Distributed integration and API management both have a decentralized architecture that deploys only the required functionality to a specific location or team. Containers serve as the underlying platform for both approaches because their immutable nature keeps images and deployments consistent across environments, so they can be rapidly deployed or replaced without opaque dependencies or conflicts.

The key to a distributed architecture—whether with integrations or with APIs—is that there has to be a way to design and deploy new services without a complex approval process.

Containers allow distributed integrations and APIs to be treated as microservices. They provide a common tooling for both development and operations teams and the ability to use rapid development processes with managed release processes.

Each container represents a single service or application, like a microservice represents a single, discrete functionality. In a microservices architecture, there can be dozens or even hundreds of separate services—and those services are duplicated across development, test, and production environments. For that number of instances, the ability to orchestrate instances and perform advanced administration tasks is critical for the container environment to be effective.

---

11 *Vasudevan, Suresh, "Containers And Kubernetes Are Powering The Second Cloud Adoption Cycle," Forbes. July 10, 2019,* https://www.forbes.com/sites/forbestechcouncil/2019/07/10/containers-and-kubernetes-are-powering-the-second-cloud-adoption-cycle/#171ce5006929

Red Hat OpenShift combines Linux® containers with Google's Kubernetes orchestration project. It includes centralized administration, such as instance management, monitoring, logging, traffic management, and automation, which would be almost impossible in an environment with containers alone.

Red Hat OpenShift also supplies developer-friendly tools like self-service catalogs, instance clustering, application persistence, and project-level isolation.

This combination balances the requirements of operations, particularly for stability and testing, with developer needs for easy use and rapid delivery.

Implementing agile integration

Team practices

The three pillars of agile integration are most effective when deployed and available to teams as reusable capabilities. What Red Hat means by "capability" is that authorized groups can use the technologies in a self-service manner, follow organizational guidelines easily, and gain access to best practice information.

Information architects or IT administrators have to define clear processes for the individual teams, such as:

- Providing widely available usage guidelines.
- Enforcing usage and best practice rules where appropriate but allowing freedom for experimentation beyond those rules.
- Having well-defined processes for moving from prototype, through test, go-live, updates, and retirement.
- Allowing information sharing for new deployments and developments.
- Using infrastructure teams as enablers and providers of self-service capabilities rather than forcing them to be part of every process.

For example, it should be possible for a software team to develop, test, and prepare a new API for launch entirely via self-service, with processes in place to update other groups and documentation.

There may be processes and cross checks with other teams before publishing or moving into production, but the infrastructure should automate the process as much as possible.

Infrastructure architecture

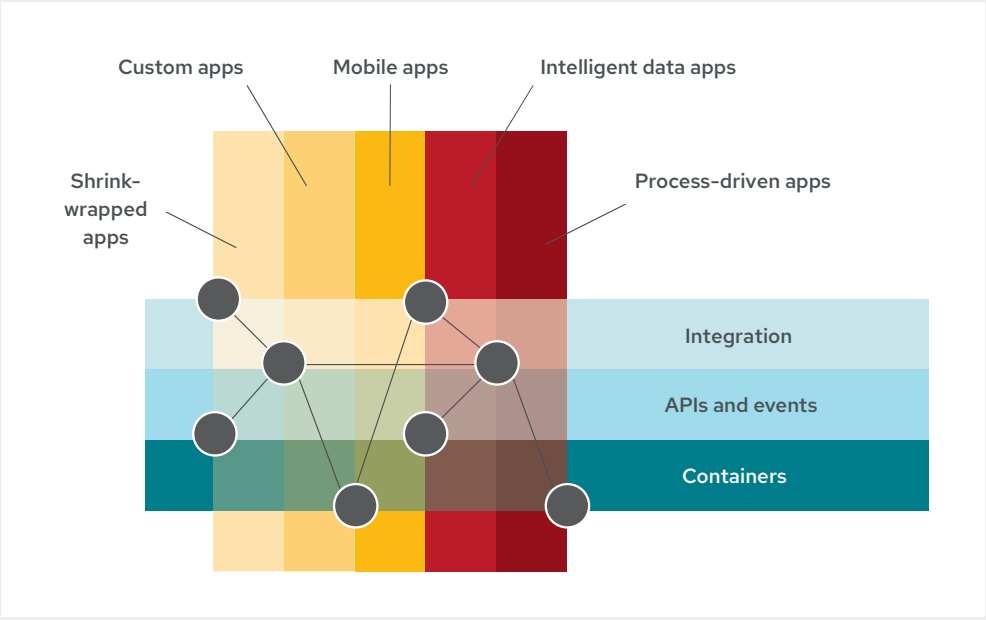

Figure 4. How agile integration technologies cut across application stacks

An organization's internal software ecosystems – and, in many cases, the access points for external integrations – are made up of two layers, a synchronous layer and an asynchronous layer. Containers, APIs, and integration work together on the synchronous layer. Events, containers, and integration work together in the asynchronous layer. Event processing includes event mesh and distributed streaming platforms.

Different types of systems expose a variety of reusable endpoints, each visible as a reusable API and many running within containers for scalability and ease of deployment. Integrations provide transformation, composition, or inline business logic wherever needed through the system by integrating a group of individual services or gathering results from different parts of the organization.

Integrated applications can be further aggregated before serving end-user applications.

There is no assumption that all systems will be decomposed into increasingly small pieces or pass through multiple layers of API abstraction. Such operations can reduce efficiency, cause latency, or add unnecessary complexity. In some areas, it may be the right choice to keep existing legacy ESB functionality to retain connections between specific applications. The dependencies between distributed systems also need to be tracked and managed using appropriate tools.

However, for the system as a whole, recasting architecture in terms of containers, APIs, and integration means the right choices can be made for each service, integration point, and customer interaction. For example, high-volume inbound requests can be security checked and then routed directly to the correct backend service, without going through a single ESB bottleneck.

In hybrid, distributed cloud environments, many of the backend systems in question may reside in different physical locations. Integrating locally proximate systems to serve a local need provides more efficiency and security than routing everything through a single central integration system that holds key business logic.

Agile organizations and culture

The life cycle of infrastructure is very different from the life cycle of software development or operations. The cycle for software development is to complete a project and then move on to the next project. Efficiency means increasing how quickly a product can be released or how many features can be produced in a given time. Even for operations, which is focused more on maintenance and stability, it is still beneficial to more efficiently and quickly apply security patches and updates, deploy new services, or rollback changes.

However, infrastructure has a very different approach. Infrastructure tends to be worked on with longer time frames and with disparate, highly specialized groups, which is very different from the cross-functional teams working on a specific software engineering project.

Infrastructure projects are typically much larger than software projects, which means that the short release cycles may not be able to accomplish much—or may leave the project incomplete. As Andrew Froehlich, an enterprise IT professional, wrote in InformationWeek, infrastructure has a point of no return limitation—especially with hardware and datacenters, but even with public cloud, there is a point where you can no longer scrap a project and start over.[12] Infrastructure is permanent. However, it is possible to reconcile methodologies with the performance of infrastructure.

The benefits of responsive, iterative processes like agile and DevOps are apparent for development and operations teams, but less so for infrastructure teams. However, it is important to align infrastructure teams with development and operations teams for maximum efficiency. As global consulting firm McKinsey found, "Using an agile transformation to modernize an IT infrastructure organization isn't easy, but it is worthwhile. In our experience, agile approaches can enable IT infrastructure groups to boost their productivity by 25 to 30 percent in six to 18 months, depending on the size of the organization."[13]

Agile integration technologies underpin a more agile infrastructure. APIs, container images, and distributed integrations become new methods of discourse in software infrastructure conversations.

---

12 *Froehlich, Andrew, "Should IT go agile? The pros and cons." October 6, 2015,* http://www.informationweek.com/infrastructure/pc-and-servers/should-it-go-agile-the-pros-and-cons/d/d-id/1322448

13 *Cormella-Dorda, Santiago, et al. "Transforming IT infrastructure organizations using agile." McKinsey Digital, October 2018,* https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/transforming-it-infrastructure-organizations-using-agile

The Agile Manifesto defines four core principles for software development.[14] In an agile, integration-based infrastructure, these principles can be applied to the integration strategy.

1	**Individuals and interactions over processes and tools.** With infrastructure, the discussion is focused on interactions between teams. Interactions include direct communication, governed by APIs, messaging, and traffic patterns; systems-level interdependencies; and testing and release process, such as CI/CD pipelines.
2	**Working software over comprehensive documentation.** Infrastructure by its nature must be functional 24/7/365, with gradual adaptation rather than major changes. In that sense, a working infrastructure is always an implied requirement. As an infrastructure strategy, "working" means that the infrastructure component delivers the expected end-user behavior in the expected performance envelope.
3	**Customer collaboration over contract negotiation.** With infrastructure systems, contracts represent how infrastructure teams manage system dependencies, such as security policies, service-level agreements, and even published APIs. Customers include both internal and external users of those systems. Agility gives those users a voice in potential changes in policies and interfaces associated with systems and lets them see those changes executed more quickly. Using distributed integrations extends that collaboration by giving control to develop and deploy integrations directly to teams.
4	**Responding to change over following a plan.** This is a principle where technology supports process. For infrastructure, the systems should remain stable, but newer technologies like containers provide a platform that is elastic. It is possible to dynamically add and remove instances according to demand, to automate deployments and updates, and to orchestrate changes across multiple instances. Published API definitions provide reusable tools to help development be more consistent. This approach makes a stable platform that is designed to adapt to change.

Figure 5. Core principles for software development, from the Agile Manifesto

Agile integration uses technology to support culture change within infrastructure teams. It serves as the foundation for infrastructure strategy. It aligns the infrastructure technologies and its teams more closely with development and business strategies.

The agile methodology identifies some key parts of a software project, such as individuals, builds, and dependencies. It can then define relationships between these elements. When approaching integration infrastructure as an agile project, there are similar elements and relationships that can be identified, paralleling those defined by agile—such as teams, container images, APIs, and integration points. Table 3 describes some of these parallels.

14 *Agile Manifesto,* http://agilemanifesto.org/

Table 3. Comparison of elements of software agile and infrastructure agile

Project	Organization	Detail
Individuals	Teams	Teams are responsible for particular parts of the infrastructure. This identifies information surrounding team responsibilities, such as the systems or APIs managed by the team, team leaders, and the goals of the team.
Modules	APIs	Well-defined interfaces (APIs) are stable over time, have their own roadmaps, are run by specific teams, and create a particular capability important within the organization.
Builds	Container images	Releases are based on deployable units that have been tested, tagged, and can be deployed dependably by any team that has access. This replaces monolithic, versioned code.
Compile dependencies	Integrations	This element identifies the integrations and mappings between different components in these distributed systems. These integration points can then be managed, commisioned, decommisioned, versioned, and tested just like any other part of the system.
Build testing	Infrastructure automation	This is full life-cycle management, from the ability to test software builds, performance, and user requirements to operating and monitoring multiple systems.

Applying agile principles to infrastructure planning

Most change management approaches require comprehensive documentation of all subsystems. This documentation has to cover, in detail, every aspect of the system, from monitoring method to performance parameters to responsible teams. Agile principles require collaboration and adaptability, which is in conflict with documentation-heavy change management.

Rather than trying to prescriptively define all potential stakeholders, changes, and system components, define a set of guidelines and standards that can be used to evaluate change requests and planning. Consider these questions:

- What is the intended end-to-end experience for the user?
- How is everybody – each team, API, and system – contributing to improving this experience over time?
- How will monitoring and alerting be defined, and with what parameters, to maintain service levels?
- What kind of automated testing is needed to verify the expected behavior?
- What is the release pipeline for teams to test and deploy new versions of their own subsystems without disrupting the user experience?
- How does a failure in a component service affect the service levels of the whole system?

Change management within an agile infrastructure should be less of a contract and more of an ongoing collaboration.

Are the odds in your favor?

How likely is your IT project to succeed? First, it depends on knowing your criteria for success—is it meeting specifications, increasing customer adoption, or just releasing it? A Project Management Institute survey revealed that more projects are meeting their planned targets than in the past five years. They attribute the uptick to stronger alignment between IT and business teams, leading to better information about strategy and customer needs.[15]

One of the reasons for that strategic alignment is implementing agile teams. Agile encourages collaboration and feedback, a holistic view of problems and systems, and creative approaches.

Having a shared technology stack moves discussions away from independent code to systems and their interdependencies. This is systems-level thinking, treating the entire collection of software infrastructure—including internally developed software, vendor systems, and the connections between them—as a single system. APIs and messaging systems can span the entire infrastructure and work to unify the software systems.

Because APIs and distributed integrations can be developed and understood within individual development or operations teams, the knowledge of team responsibilities for integrations is much clearer. The integrations themselves are better understood because the interdependencies between systems and applications are recognized by the teams handling the development and deployment.

Using integration as the foundation for infrastructure, and then distributing responsibility for that integration across teams, creates an infrastructure environment where agile approaches are more relevant.

15 *Florentine, Sharon, "IT project success rates finally improving." February 27, 2017.* https://www.cio.com/article/3174516/project-management/it-project-success-rates-finally-improving.html

Conclusion: Delivering agile integration

Agility is a process, not a project.

It has never been more important for organizations to be able to react to change in the market, and IT systems must deliver this ability to launch new services or update existing ones quickly. Rethinking IT infrastructure is crucial, as it is the foundation of digital services.

Infrastructure teams have historically been tied to very long, modulated processes because of the need to mitigate risk and maintain stability. However, it is possible to shift the mindset of infrastructure from hardware or platform-based to integration-based. Integration is not a subset of infrastructure. It is a conceptual approach to infrastructure that includes data and applications with hardware and platforms.

Red Hat defines this approach as agile integration, a way of using the three pillars of agile integration—distributed integrations, APIs and containers—to create a more agile and adaptive infrastructure. Technology has to be used to support culture change, and that means working to make infrastructure teams—and not just their software—more agile. As infrastructure teams work to align themselves with agile principles, technology can gradually be introduced to support those changes. There is no single project that will rearchitect an entire organization to be agile. It may be more effective to implement one agile integration technology or change one area of the business and then extend those changes incrementally.

Improving the responsiveness of IT infrastructure to change is a long-term strategic goal. Sweeping, organization-wide changes do not need to be made for there to be progress. It may not even be necessary to try to make changes in isolation and then roll them out.

Agile integration provides a framework, both technical and organizational, to help reshape IT infrastructure.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
#F20123_1119