AGILE INTEGRATION:

THE BLUEPRINT FOR ENTERPRISE ARCHITECTURE

E-BOOK

by Steve Willmott and David Codelli
Edited by Deon Ballard

TABLE OF CONTENTS

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

A new architectural framework called agile integration brings together three important architectural capabilities: containers, distributed integration, and application programming interfaces (APIs). This framework addresses how these key capabilities promote agility and power new processes within your organization to create 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.

To stay relevant, organizations need the ability to plan and execute changes to their software systems quickly.

For software delivery at today's speeds, organizations need an agile infrastructure foundation. For this definition, agile does not refer to agile software development. It refers to the more traditional meaning of agile—flexible, able to move quickly.

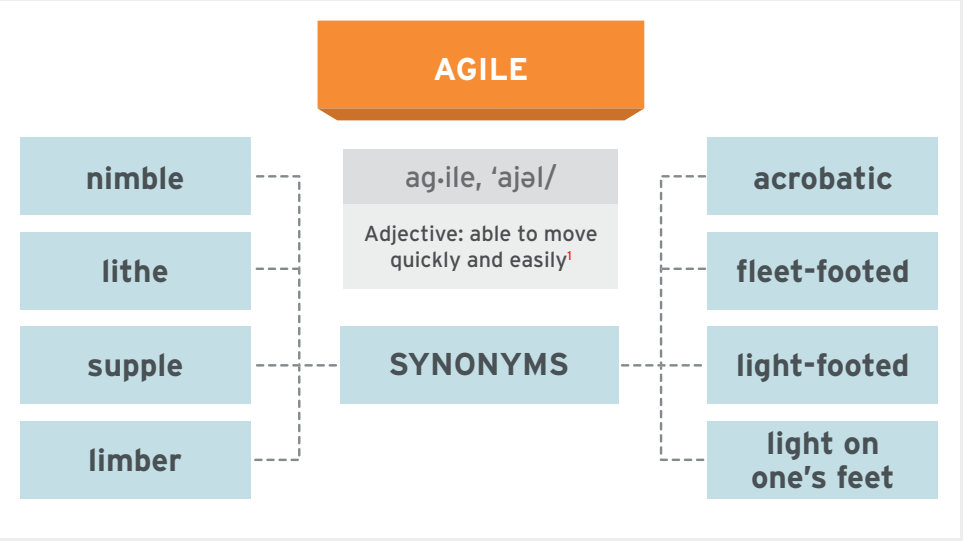

Figure 1. Definition of agile

1 Oxford English Dictionary

"The demand for agility to continuously win, serve, and retain customers requires the interfaces between systems of engagement and systems of records to become more agile—agile in terms of scalability but also in the ability to adapt quickly, for example, to add a new attribute on existing APIs and for the future to provide more context."

HENRY PEYRET
THE FORRESTER GROUP

Peyret, Henry. "TechRadarTM: Integration Technologies, Q2 2015." Forrester Research, Inc. June 23, 2015.

To date, agile methodologies have focused on software development, trying to improve and streamline how applications are created. DevOps[2] practices have tried to carry that methodology into how these applications are deployed.

DevOps itself, however, generally only reaches so far, addressing primarily new software applications developed by the organization itself.

Infrastructure agility goes even further and creates an environment that encompasses all IT systems, including legacy software. An agile infrastructure is an approach that takes the complexity of existing systems, different data types, datastreams, and customer expectations, and finds a way to unify them. This is, at its heart, an integration problem.

An organization that can change pricing overnight or make new products available for sales overnight has an enormous advantage over one that requires a three-month staged roll out with a cascade of manual verification steps.

We call this 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, it is possible to create a platform that provides your teams with the ability to change as quickly as the market demands.

PLANNING IS DEAD: ORGANIZATIONS AND AGILITY

"Planning as we know it is dead," was the keynote message delivered by Jim Whitehurst, Red Hat CEO, at the 2017 Red Hat Summit. "Planning in a less-known environment is ineffective."[3] 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 you have, or the less stable your environment is, the less valuable your plans are.

You don't know what you don't know

Infrastructure planning typically takes a long-term approach, sometimes spanning years. Trying to create a multi-year plan can strangle the ability to innovate or pivot as the market changes. The "death" of planning alluded to by Jim Whitehurst comes down to the ability to plan more quickly and then execute on those plans. It is a shorter life expectancy for plans and an environment that cultivates new plans.

This rapid change can be challenging when teams are accustomed to 6-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.

2 *Innovate faster with DevOps* https://www.redhat.com/en/insights/devops

3 *Jim Whitehurst keynote address at 2017 Red Hat Summit.* https://www.cbronline.com/news/enterprise-it/software/red-hat-ceo-planning-know-dead/

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 freedom they have to structure their infrastructure, teams, application, architecture, and even their deployment processes. It is more than 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."[4] They can be agile.

Beyond the ability to build something new, these organizations also build systems that are ready for 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 start-ups age, some suffer from a reduced ability to adapt, but the best organizations ensure their ability to change is protected at all costs.

Rising to the challenge

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

Change needs to occur at two levels:

- Organizational and cultural support of agile processes—from architectural design to team communication.
- Technical infrastructure that creates the ability 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.[5] This makes new projects and innovations a priority.

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

5 Cagan, Marty, "Inspired: How to Create Products Customers Love." Wiley Press, 2017

"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, 10% 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."

GENE KIM
THE PHOENIX PROJECT

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.

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.

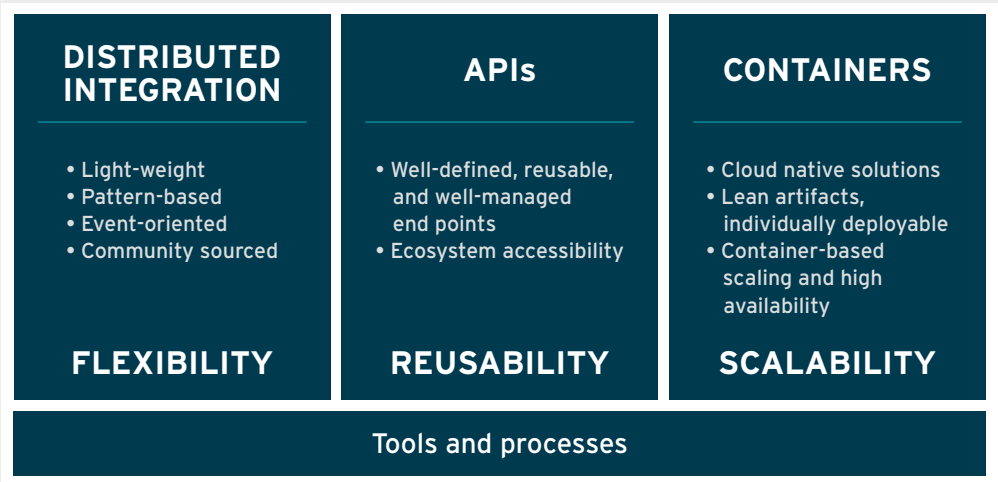

Figure 2. Three pillars of agile integration

1. **Distributed integration.** A few dozen high-level integration patterns reflect enterprise work and dataflows. When these integration patterns are deployed within containers, the integration patterns can be deployed at the scale and location needed for specific applications and teams. This is 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, allowing those interfaces to work as building blocks for reuse 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 your integration platform enables a much more transparent and collaborative relationship between development and infrastructure teams.

These three technologies 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.

Distributed integration

One of the biggest challenges of current IT systems is that they need to connect applications from across organizations. The difficulty of integration services has 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 for rapid change.

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. 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 affecting any other integrations deployed throughout the organization. This distributed approach allows the flexibility that agile projects require. It also uses the same toolchain as the agile or DevOps teams by using the underlying container platform, increasing the ability of teams to manage their own integrations with their own tools and schedules. This essentially treats integration as a microservice,[6] which increases the speed of development and release integrations.

Alignment with developer tools and processes is critical. A core aspect of distributed integration is that it 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.

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

> *"In software, when something is painful, the way to reduce the pain is to do it more frequently, not less."*
>
> **DAVID FARLEY**
> CONTINUOUS DELIVERY: RELIABLE SOFTWARE RELEASES THROUGH BUILD, TEST, AND DEPLOYMENT AUTOMATION

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

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

To use an ESB, a team is forced to use that ESB's tools for the entire life cycle, in addition to whatever tools are being used in the development and operations environments. This limitation leads to awkward, inefficient, and error-prone operations.

Messaging strengthens integration

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

Integration technology needs to be able to support this kind of 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 JBoss AMQ to provide a messaging infrastructure. A strong messaging infrastructure ensures events and data are routed between systems effectively. Messaging is an important architectural tool with microservices because its asynchronous nature requires no dependencies.

This combination of integrations and messaging improves the overall performance of the integration architecture by offering more effective routing, support for multiple languages and protocols, asynchronous throughput, and better data management.

"A new competitive rivalry, often referred to as digital transformation, is driving the need for organizations to rethink their IT architecture; redistribute workloads across on-premises infrastructure, clouds and things; and interoperate to support business strategy and operations. All of these changes require a new approach to integration – an approach we refer to as 'hybrid integration.'"

CARL LEHMANN
451 GROUP

Carl Lehman, 451 Research, "The Disruptive Role of Integration PaaS and APIs in the New Hybrid Integration Platform Market." July 2017. https://451research.com/report-long?icid=3862.

Following a trend

Container adoption is growing—but by how much? And why? 451 Research predicts a 250% growth in the market[7]—but that's in spending, not deployments. Actual deployments are a little harder to gauge. A Bain survey commissioned by Red Hat found about 20% of customers currently are deploying containers in production, and roughly the same amount in development and test environments—but over 30% were evaluating containers or running proofs-of-concept.[8]

Part of the murkiness lies around what it means to use containers. The Enterprisers Project outlined four different patterns for container adoption: using it as a general development or deployment platform, using it as a cloud-native or microservices platform, using it within a hybrid cloud, or using it for innovation projects.[9] How you are using containers can influence how you view its adoption.

For agile integration, the idea is to create an infrastructure platform that can support existing operations. The platform may borrow from all of the implementation patterns, but at its core, it works as a platform—a foundation for both new projects and existing services.

Containers

Virtualization, cloud, and containers are similar technologies that try to accomplish a similar goal. These technologies abstract the operating environment for software away from the physical hardware so that it is 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 outlined by container technology is what has made 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. This makes the environment highly repeatable and consistent for each instance, which is ideal for continuous integration and continuous delivery (CI/CD) pipelines. Additionally, because a container image only defines what is needed for a single application, containers are matched with microservices, and container orchestration can also orchestrate the deployment and management of large microservices infrastructures.

The combination of lightweight and repeatable makes containers an ideal technology platform for agile integration.

7 *451 Research infographic based on the Cloud-Enabling Technologies Monitor report, January 2017.* https://451research.com/images/Marketing/press_releases/Application-container-market-will-reach-2-7bn-in-2020_final_graphic.pdf

8 *Bain survey: For Traditional Enterprises, the Path to Digital and the Role of Containers, November 2016.* https://www.redhat.com/en/resources/path-digital-containers

9 https://enterprisersproject.com/article/2017/8/4-container-adoption-patterns-what-you-need-know

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 work as the underlying platform for both approaches because their immutable nature keep images and deployments consistent across environments so they can be rapidly deployed or replaced without opaque dependencies or conflicts.

The key of 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.

Containers require orchestration

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 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.

Red Hat OpenShift combines Docker containers with Google's Kubernetes orchestration project, and 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.

APIs

Most information infrastructures contain hundreds or even thousands of systems, applications, and assets, but 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 are the interfaces for all of the assets that can be connected using integration technology. APIs are a set of definitions or rules that set up how applications communicate with each other.

As organizations shift from a centralized, integration technology center-based 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 their companies. A strong API capability delivers this authority and autonomy to those teams. With APIs, the teams get the integration they need while the organization can ensure that security, authorization, and usage policies are managed and enforced. APIs provide reference to teams on how integrations can be designed.

APIs are different from a final application. They determine how applications can interact, and then individual developers use them as building blocks within their projects. APIs give developers and teams a common language. An organization can even use their APIs to foster communities that share and cooperate to create new, innovative uses for the services.

Different APIs, or different subsets of an API, can be made available to different audiences. The needs of a vendor may be different from those of internal development teams or community developers. 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.

As with distributed integration, containers can serve as a platform to develop, deploy, and manage APIs in ways that align API development with larger development and operations processes and tools.

The right API platform can help your developers do more

The power of APIs comes from the ability of others to use those APIs—both internal developers and external users. Red Hat 3scale API Management Platform provides tools to help all users. It provides a developer portal for developers to collaborate on creating APIs, and an admin portal to be able to publish those APIs.

3scale API Management Platform also helps make those APIs consumable externally by providing authentication, integrating with major cloud providers, and running inside containers.

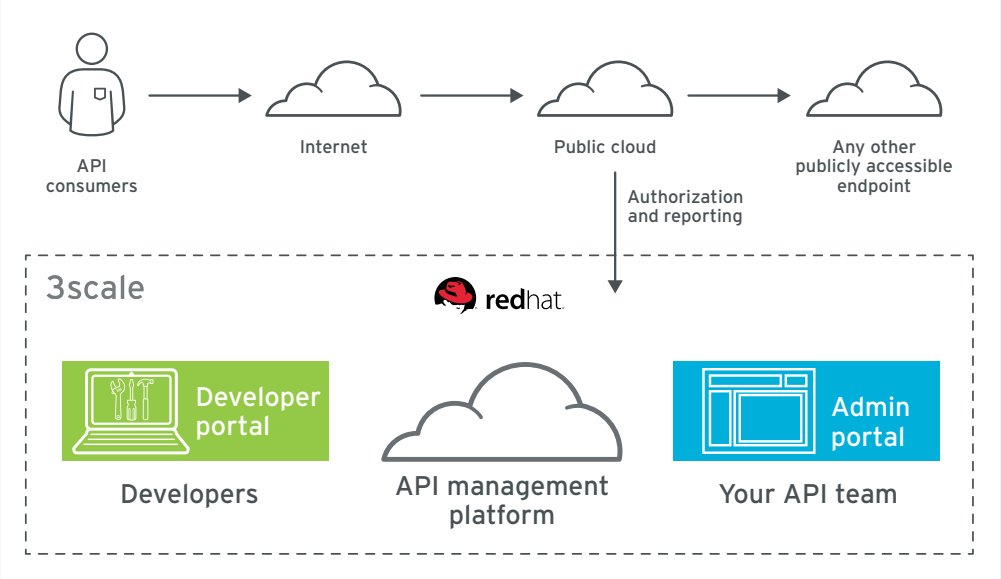

Figure 3. A view of API management, endpoints, and the public cloud

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

ARCHITECTURE OF AGILE INTEGRATION

Team practices

Agile integration pillar technologies are most effective when deployed and available to teams as reusable capabilities.

What we mean 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 in an entirely self-service way, 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

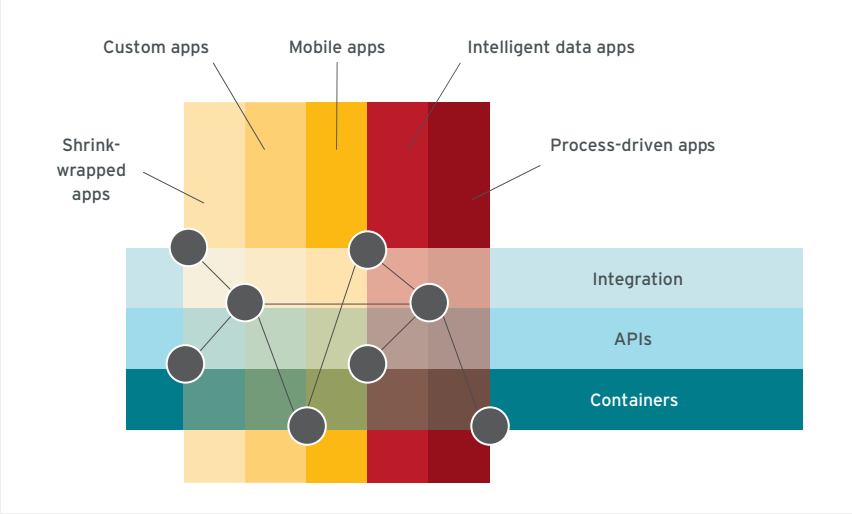

Figure 4. How agile integration technologies cut across application stacks

Containers, APIs, and integration act together to provide solid foundational layers for an organization's internal software ecosystems—and, in many cases, the access points for external integrations.

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.

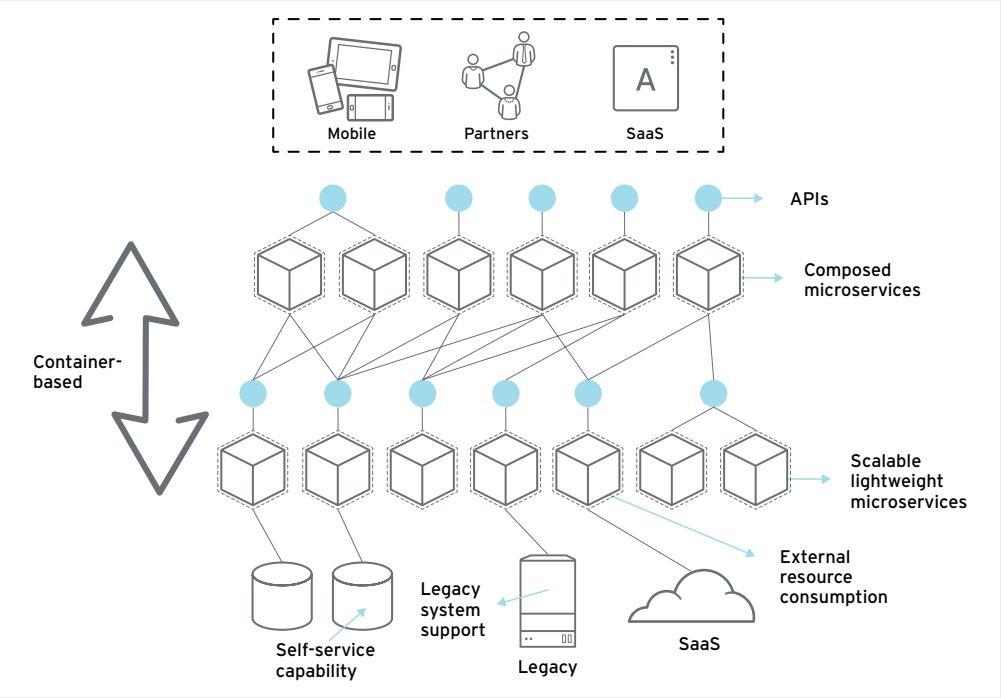

Figure 5. An infrastructure design with containers, APIs, and distributed integration

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, add latency, or add unnecessary complexity of their own. 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 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 apply security patches and updates, deploy new services, or rollback changes more efficiently and quickly.

However, infrastructure has a very different approach. Infrastructure tends to be worked on at longer timeframes 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 things incomplete. As Andrew Froelich, 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.[10] 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, Froehlich's pro/con analysis of agile for infrastructure misses one critical aspect: aligning infrastructure teams with development and operations teams. Rohan Pearce wrote in CIO about changing infrastructure teams into agile-style work cells rather than functional teams.[11] Telstra Enterprise Services teams were having developer groups simply ignore their internal systems because the process to check out systems or make updates was so painful and complex. Adjusting their working groups reduced cycle times from 212 days to 42 days.[12]

This example illustrates how critical process change can be for infrastructure teams to more effectively serve their internal groups.

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

---

10 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

11 Pearce, Ronan, "Can infrastructure be agile?" June 20, 2013.
https://www.cio.com.au/article/465436/can_infrastructure_agile_/

12 http://agilemanifesto.org/

The Agile Manifesto defines four core principles for software development.[12] 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 6. 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.

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, commissioned, decommissioned, 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 for 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? Project management training group 4PM defines success as completing a project on budget, on time, and to specifications.[13] With that definition, they estimate about 70% of IT projects fail.[13] Those numbers are starting to shift. A recent Project Management Institute survey revealed that more projects are meeting their planned targets than in the past five years.[14] They attribute the uptick to stronger alignment between IT and business teams, leading to better information about strategy and customer needs.[8]

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.

13 *4PM.com, "Why projects fail so often." September 27, 2015.* *http://4pm.com/2015/09/27/project-failure/*

14 *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*

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 the 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.

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 is largely IT systems that must deliver this ability to launch new services or update existing ones quickly. Rethinking IT infrastructure has never been more important, 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.

We define this approach as agile integration, a way of using integration technologies to help create a more agile and adaptive infrastructure. There are three technology pillars to agile integration:

- **Distributed integration,** which uses messaging and enterprise integration patterns to integrate data and systems. These are broken down into small, team-driven integrations that are distributed, as needed, across projects and touchpoints.
- **Internal API management,** which creates a reusable set of interfaces to allow development teams to engage with applications and systems. APIs provide guidance and structure to how applications should interact.
- **Containers,** which allow integration projects to be closely aligned with development and operational projects and enable integrations to be developed, tested, and released similarly to software projects using DevOps methods.

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.

ABOUT RED HAT

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

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

redhat.com
f11423_0518

NORTH AMERICA
1 888 REDHAT1

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