E-book

Red Hat

Cloud-native meets hybrid cloud: A strategy guide

Building application environments for reliability, productivity, and change

Table of contents

Introduction

Information technology (IT) systems are the glue that holds modern enterprises together. They power internal systems, are often critical market differentiators, and need to deliver ever-improving functionality for customers, employees, and partners. At the same time, system complexity and speed of change have escalated: deployments routinely span multiple on-premise and cloud locations, an array of software technologies are used, and the teams behind these systems can vary greatly in their skills. The increased focus on digital transformation highlights the business advantages of change, but often does not address how to deliver real value with the new technologies.

These pressures lead to a fundamental tension between the reliability and productivity of the environment. On one hand, systems need to be kept functioning, secure, and predictable in their operation. Developers and operations teams need to be able to evolve the systems rapidly to deliver new functional benefits to users so the whole organization benefits from progress.

This core tension between reliability and productivity impacts many choices, but it becomes particularly problematic given today's pressure for rapid change and improvement. IT teams find themselves continually having to balance short and long-term change in application delivery and development. For example, trading the use of existing systems and infrastructure to develop new functionality versus redesigning development and delivery capabilities. The long-term aim is to keep adding value with the company-wide application environment, while improving agile development and delivery.

Managing these pressures is relevant since IT systems have become highly distributed across datacenter locations and public clouds—hybrid cloud IT is a reality for most organizations. In addition, a myriad of new development technologies in cloud-native and other areas open up huge new opportunities to make development teams more productive.

This guide is designed as a strategy primer for enterprise architects and IT leaders to map out how to tackle modern IT strategy. It is focused on the convergence of three trends:

1. The importance of IT systems to modern organizational success.
2. Hybrid cloud and the shift to multiple datacenters, platforms, and cloud locations.
3. Cloud-native application development technologies and how they can be combined with existing approaches to create a robust, productive IT system.

The content covers context considerations, strategy challenges, and high-level architectural considerations with case study examples.

"Like many of our customers, we faced the real questions of, 'How can we do better than just keep up?' and 'What can we do to be well prepared for whatever comes our way?"

Mike Kelly
CIO, Red Hat

This e-book:

- Focuses on strategic aspects and high-level architectures, specifically, high-level concepts that can be applied in a wide variety of contexts. It does not address detailed deployment considerations. Resources for deeper technical analyses are offered throughout.
- Does not assume greenfield development only. While some teams may have the luxury of building new systems from scratch, the majority of our advice applies to the evolution of complex existing systems.
- Provides general technical applicability. Red Hat® customer and technology examples are used in a variety of areas. While we believe Red Hat tools are right for the job, most approaches discussed in this e-book can be applied in other technology environments or using a mix of Red Hat and other technologies.

Companion content can be found in the following e-books: Teaching elephants to dance, The path to cloud-native applications, Principles of container-based application design, and a range of others available from the cloud-native resources page at redhat.com.

Part one: Modern IT challenges

Early in the summer of 2017, Mike Kelly, the newly appointed Chief Information Officer (CIO) of Red Hat was in a difficult position. Red Hat had a robust IT infrastructure that had grown tremendously, but even more growth was approaching. Red Hat's rapid growth meant that the IT team needed to get ahead of the growth curve while still operating within a constrained budget.

Red Hat Innovation Labs is an immersive teaming residency program for customers to accelerate their most innovative ideas.[1].

How did Mike and his team approach these challenges?

Transforming Red Hat IT

IT systems can be thought of as the central nervous system of an organization, carrying information, coordinating actions, and executing many of the tasks that enable the organization to function. The size, density, and complexity of this system of applications has increased for most organizations in recent years. Red Hat was no different.

By 2016, Red Hat operated nearly 1,000 unique, independent applications and services that served various parts of the business. These applications were run by different teams, on different technology stacks, and across a number of redundant datacenter locations. The objectives for the team was a three-part challenge:

1. Match the speed and adaptability demands of digital business.
2. Improve the availability, resiliency, and security of digital systems.
3. Continue to reduce operating costs.

The objectives show typical dilemmas: The need for speed, adaptability, availability, and security—but with lower operating expenses.

The applications powering Red Hat's day-to-day business ranged from custom JavaTM and other applications, to integration of Software-as-a-Service (SaaS) and data stores, to complex workflow processes.

1 *Red Hat Open Innovation Labs, 2019. redhat.com/en/services/consulting/open-innovation-labs.*

The project begins

To get started, Red Hat IT engaged the Red Hat Open Innovation Labs team for the thorough analysis routinely done for Red Hat customers. Red Hat's Open Innovation Labs program provides immersive residency programs to help organizations solve real business challenges with a combination of technology and process changes. The result for Red Hat IT was comprehensive insight into the choices they faced and options to address some of the most pressing issues.

The resulting program included changes—some immediate and some gradual—across a broad range of areas in Red Hat's IT processes:

- A shift from a multiple-location, on-premise failover infrastructure to a hybrid cloud infrastructure, including first one, then two public cloud zones from different providers.
- A move to a container-based, cloud-native deployment and application development mode, with containers as the primary unit of deployment.
- Significant shifts in software development technologies and methodologies to allow a wide variety of programming languages to be used according to the needs of each application.
- Significant cost savings from containers and the ability to move from a parent-child failover setup to failover between three active sites for greater reliability and faster updates.
- Upgrades to a range of applications and the increased adoption of application programming interfaces (APIs) and agile integration are a way to connect these applications.

Observable, measurable outcomes

By early 2019, the program had exceeded its original transformation goals (see Figure 1). The journey is not over, and the team continues to evolve Red Hat's IT environment as new challenges and opportunities arise. But the team's strategy allowed them to create a flexible and reliable application environment for the company. They saw a 55% lower footprint per application, improved security, and multisite hybrid cloud deployment with zero downtime. Automation of pipeline delivery gave them a 40% faster feature delivery time- to-market.[2]

Figure 1. Red Hat IT gains from its cloud-native hybrid cloud transformation (source Red Hat IT internal data)

2 *Red Hat IT internal data*

Even though more than 50% of organizations have implemented some level of DevOps practice,[3] the most committed (organizational) adopters see much higher performance gains. DORA's 2018 State of DevOps survey highlights how adoption often leads to slowdowns before it leads to gains.

There are now numerous ways for IT team members—and anyone in the company—to deploy new applications and share access to others. Infrastructure for production systems is resilient and lower in cost.

The team also learned to navigate the internal cultural challenges inherent in change. It turned out to be just as important to value inputs from skeptics on the team, as those from the enthusiastic technology adopters.

Read more about Red Hat IT's transformation.

Fast or stable?

As the Red Hat IT story shows, new technologies can provide genuine benefits to addressing modern IT challenges, but their application takes thought and coordination. The question of how to navigate short- and long-term change is particularly acute in large organizations since development teams need to move fast and adopt new and more productive technologies, while operations teams are responsible for reliability and stability.

The DevOps movement has made this conflict visible to different groups within the organization. However, even with DevOps adoption, there are typically varying practices in every organization. The challenge is exacerbated by a wide range of high-pressure trends that diversify technology choices:

- The vast majority of enterprises (84% according to a recent Flexera/Rightscale report[4]) are pursuing a multicloud strategy. In many cases, organizations have become hybrid cloud by default, simply because different groups adopt different cloud providers.
- Programming languages continue to explode in use, both in general and in the enterprise. While Java and Javascript still dominate in the enterprise, other languages are gaining traction in specialized applications. (For example, see Red Monk's 2019 public data survey.[5])
- The adoption of cloud-native technology, and particularly the move to containers and container management as infrastructure, has accelerated.[6]

The challenge is how to balance these pressures.

Creating and evolving an effective hybrid cloud application environment

We use the term application environment to refer to an organization's set of capabilities for application delivery and development. This broad view is deliberate and it underlies the holistic nature of the modern IT challenge. Specifically, legacy and new applications need to work together, new technology needs to add value without disrupting reliability, and IT systems are found in ever more locations.

An organization's application environment undergoes constant change as new applications are deployed, tools and processes are updated, and applications generate data and transactions. Ultimately, the production applications that the environment supports are what add value to organizations employees, partners, and users.

---

3 *Stroud, Robert. "2018: The Year Of Enterprise DevOps." Forrester blog, October 17, 2017, https://go.forrester.com/blogs/2018-the-year-of-enterprise-devops/.*

4 *Flexera, "RightScale 2019 State of the Cloud Report from Flexera." 2019, https://info.flexerasoftware.com/SLO-WP-State-of-the-Cloud-2019.*

5 *O'Grady, Stephen. "The RedMonk Programming Language Rankings: January 2019." Red Monk blog, March 20, 2019, https://redmonk.com/sogrady/2019/03/20/language-rankings-1-19/.*

6 *Hippold, Sarah. "Gartner identifies key trends in PaaS and Platform Architecture for 2019." Gartner press release, April 29, 2019. https://www.gartner.com/en/newsroom/press-releases/2019-04-29-gartner-identifies-key-trends-in-paas-and-platform-ar*

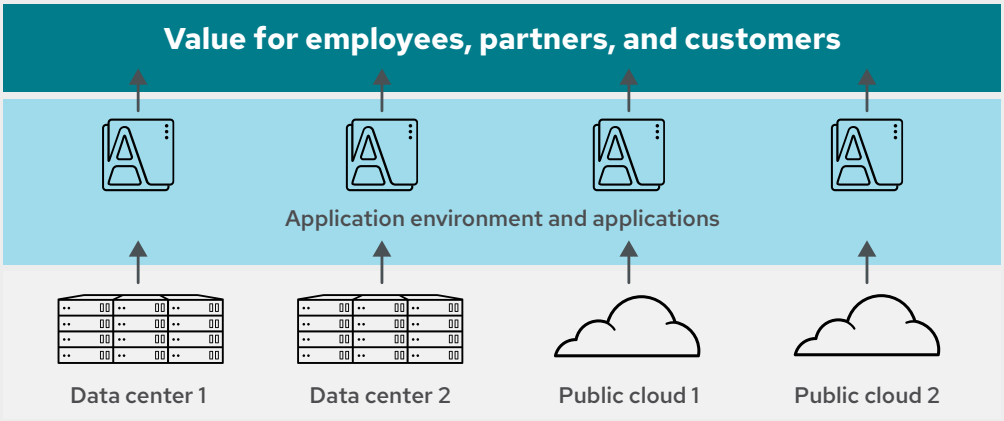


Figure 2. Application environments support applications that deliver value

Thinking of an organization's application environment as a single, organization-wide, evolving ecosystem is a powerful metaphor since it brings into focus the properties the environment needs to exhibit and what tensions are in play as individual decisions are made.

Many of the compromises mentioned in later sections, including reliability versus productivity, predictability versus the ability to change, security versus convenience, performance versus cost, and rigidity versus freedom look like an either-or proposition. They look like choices to be made in opposition to one another. While tradeoffs need to be made, for an organization to excel, IT needs to deliver on both choices:

- Reliability and productivity.
- Predictability and ability to change.
- High performance and low cost.

With deliberate, strategic planning, and focused decisions, it is possible to deliver on many of these needs.

Part two: Strategy primer

There are many different approaches to strategy frameworks. For our purposes, we adopt a simple three-step framework based on Richard Rumelt's "Good Strategy Bad Strategy: The Difference and Why It Matters".[7] The three steps provide a broad overview of what types of decisions make up a long-term strategy. A number of Red Hat internal teams use a version of this framework for planning.

Based on this model, a strategy discussion is divided into three areas:

1. **Assess the situation and set objectives.** Capture the now, the challenges and opportunities presenting themselves, as well as significant constraints on action. It would be typical to include both a long-term "where do we want to be in 5-10 years?" element and a short-term "what's coming in the next 12-18 months?" view.

7 *Rumelt, Richard P. Good Strategy, Bad Strategy: The Difference and Why It Matters. New York: Crown Business, 2011.*

2. **Set policies and guidelines.** Capture broad principles and rules of engagement that determine the overall direction of the strategy, what is in and out of scope and general practices for how certain things are done. Policies and guide rails are typically things that are durable in the midterm: at least a single year, but preferably four-five years.
3. **Determine what to do next.** Create a set of specific actions to take right now that bring the organization closer to the objectives, and more in line with the established policies and guidelines.

In Part two, we cover the first two areas of this structure. The third area is covered in depth in Part three (architecture) and Part four (where to start).

As you will see in later sections, one of the recurring patterns for success is the use of decentralized authority. As a result, it's likely that any strategy will be most effective as a multilevel construct with broad policies and guide rails at the organizational level, with leeway for individual groups to add their own local instantiations to suit their own needs.

Situation assesment and objectives

In capturing a situation assessment of your organization's IT environment, there are a number of things to consider, including from existing policies, personnel, and projected usage. The critical items will depend on each particular organization, but at a high level of abstraction there are a few elements that are important to consider regarding the overall application environment.

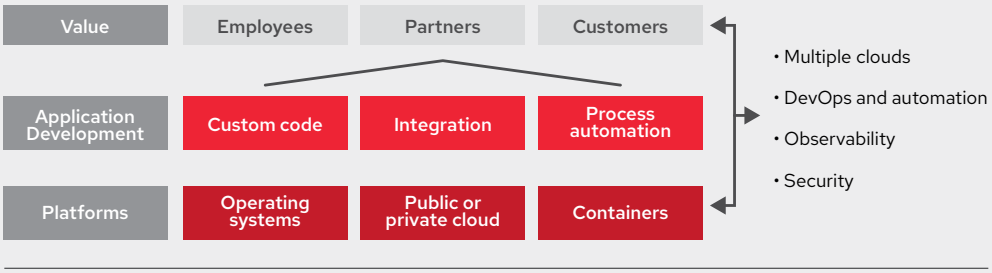


Figure 3. Simplified IT landscape view

Specifically, every organization is likely to have:

- **A set of on-premise datacenters—infrastructure hosted with one or more private hosting services, and public cloud providers.** The platforms are often distributed across geographies for failover or data protection. Regions often comprise one or more types of operating systems, virtualization systems, private cloud deployments, or containerized infrastructure.
- **A set of applications and application development solutions.** From an application perspective, the set is likely to include custom code written by the organization or third-party consultants, commercial off-the-shelf (COTS) solutions, SaaS solutions, and a range of other applications. In addition, the organization likely has application servers or other technology to host custom code, integration solutions, messaging, and process management to connect individual applications together.
- **Developer tools, processes, automation and management capabilities** that manage different parts of the infrastructure.

- **A range of stakeholders** who depend on the application environment to be productive, including developers and operations teams who create and operate applications, and employees, customers, and partners who use the resulting application.

Sketching out these major components is a useful exercise since it provides a view of the breadth and depth of the whole application environment.

In terms of objectives, we will focus on two near-universal areas of objectives: reliability and productivity.

However, there may be other high-priority areas specific to your organization that need to be considered. We expect some of the guidance that follows will address other common areas as well, but we focus on reliability and productivity since they are common across organizations.

What we mean by "productivity"

For our purposes, productivity refers to the entire organization and all of its employees. How much is getting done throughout the whole organization? In practice, we focus primarily on the productivity of the IT, development, and operations teams responsible for improving the applications available to their end users. However, by extension, these application improvements are a large part of what delivers productivity in the remainder of the organization. There are areas, as we will see in Part three, where the definitions between developers and users are becoming blurred.

Set clear objectives

We encourage thinking through three types of objectives:

1. **General outcomes.** What do we want the application environment to do better?
2. **Common patterns and anti-patterns.** What are examples of successful applications or teams that we'd like to replicate? What are examples of failures to avoid?
3. **Gains in specific projects or initiatives.** What specific activities, such as new features, requests for new technology, and retiring old functionality, are underway but might benefit from a shift in direction?

Considerations for policies and guide rails

Once objectives are set, the challenge becomes creating processes that will lead to achieving goals. Generally, this works best by establishing a set of well-informed, clear policies and guide rails that different groups in the organization can agree upon and follow.

The precise set of policies will be different for each organization. In this section, we cover a number of policy area considerations that we've seen work in large IT environments. The set covered here is a distillation of our experience working with a wide range of customers worldwide and an adaptation of prevailing wisdom from the field that has proven useful in practice.

The policy areas covered are applicable at the organizational level. Although they draw on some of the advice already available for cloud-native and hybrid cloud strategy advice (for example, see Teaching elephants to dance) and many of Martin Fowler's blog posts on architecture or microservices), they are more abstract. Later sections will provide technology-focused advice on implementation.

"Don't make 1,000 small decisions when one big decision will do. A good 'guiding policy' channels action in certain directions without defining exactly what shall be done."

Richard Rumelt
author, adapted from Good Strategy, Bad Strategy[8]

Decentralized authority

The trend towards decentralization in system design has been prevalent for a number of years, and in technology, it is most strongly expressed in methodologies such as microservices. This architecture makes sense for many types of systems, and it is one of the areas where cloud-native technology has provided major breakthroughs.

Before considering implementation, there is another level at which decentralization can play a key role.

Decentralized systems are not appropriate for every part of an IT system, but decentralization in organizational terms is almost always beneficial. It is more important to structure responsibility boundaries among teams than code boundaries. This approach may seem like a straightforward invocation of Conway's Law; however, it is often done incorrectly in organizations as they rush to implement decentralized application development models, such as microservices.

We express this policy principle as: **Centralized authority -> Decentralized authority**.

Enable rather than own

When centralized IT empowers other teams throughout the organization and shares the responsibility for development and operations, it works better than trying to centralize authority.

"Leadership is embedding the capacity for greatness in the people and practices of an organization, and decoupling it from the personality of the leader."

David Marquet
Submarine Commander, U.S. Navy[9]

This doesn't mean that central teams devolve all responsibility. There are key processes and systems they must require other teams follow. However, by providing the means for others to take responsibility, groups are much more engaged in making the whole organization function. This leads to:

- Reliability gains and better sharing of best practices (smaller teams may unearth problems before central teams).
- More individual investment in operational outcomes.
- Clearer supervisory focus on what is important versus what is not.

Productivity gains include:

- Better ability of local teams to rapidly adapt regulations to their context.
- Faster decision making for smaller items, resulting from a greater sense of autonomy and ownership.
- More pride in execution and sense of purpose, leading to higher levels of dedication and even better productivity.

Decentralization may be enabled by technology, via remote work, efficient ways of tracking decisions, and organizing teams, but it is first and foremost a cultural issue. Small shifts in directives and responsibilities can lead to significant differences in outcomes. More treatment of cultural issues can be found in Part four, and a closer look at experiences from Red Hat's innovation culture can be found in the *The Open Organization*.[10]

---

8 *Rumelt, Richard P. Good Strategy, Bad Strategy: The Difference and Why It Matters. New York: Crown Business, 2011.*

9 *Marquet, L. David. Turn the Ship Around! A True Story of Turning Followers into Leaders. New York: Portfolio, 2013*

10 *Whitehurst, Jim, The Open Organization. Boston, Massachusetts: Harvard Business Review Press, 2015.*

Red Hat

Universal capabilities

Increased numbers of public cloud deployments and datacenters can result in a rapid proliferation of new segmentation in an organization:

- Certain services only available on Microsoft Azure, others on Amazon Web Services (AWS), and others only in the corporate datacenter.
- Certain services only available in specific areas of a hosted system controlled by a specific group.

Some of this separation exists for good reason—security firewalls or data protection—but it often involves an extremely complex process to determine where to host a new piece of functionality. Difficult learning curves on how to do the same task in different environments, and other dysfunction, are not far behind.

The key policy shift to consider is a move from the typical minimalist approach to solving a challenge within a single group of: "We needed capability X, we selected or built it in environment A, and if you need to use it, it's there." To one of:

"We needed capability X, we selected or built it and made it available in environment A. If required, it could be rolled out with the exact same interface or configuration in environments B, C, and D."

In other words: **Single-use functionality -> Universal capabilities**.

This concept seems simple and obvious, yet the forethought required is only done briefly, if at all.

As a general principle, capabilities in a hybrid cloud environment should:

- Be available in an identical form in as many datacenters and cloud locations as in which they may be needed. They should use the same APIs, endpoints, and configurations whenever possible.
- Have distributed capabilities such as messaging, integration, logging and tracing, and process automation that run seamlessly across datacenter and cloud locations, and transparently connect those locations.

Note: Depending on the service, being "available" in another datacenter or cloud may mean that either the same instance is simply accessible as a service of an API remotely from other locations, or that local instances of the same service are available.

By insisting on interface uniformity, and the ability to recreate capabilities in the exact same way across locations, it is much easier to create failover configurations. Doing this reliability prework as new capabilities are planned and configured has a dramatic effect later in the process.

This approach also leads to a uniformity of execution and reuse that delivers greater benefit than using different underlying capabilities in different locations. From a developer productivity point of view, having the same set of available services reduces the amount of learning required and increases the applicability of skills learned by the team.

(Some) Constraints = Freedom

The desire to empower a greater number of teams in the organization to contribute and operate autonomously is fine in principle. But how does it affect the system in practice? The most important points becomes evident in terms of control versus freedom: "You tell us we're responsible and autonomous, yet you impose a huge raft of restrictions and technologies."

- Too little enforcement can lead to a lack of rules and experiments that create fragile application environments.
- Too much enforcement tends to lead to resentment, disengagement or frequently creative workarounds that end up being even more fragile. These workarounds often contain useful ideas but end up impeding long-term progress.

To succeed, the question is not "how restrictive should I be," but "where should I be restrictive and where should I be loose?"

A useful illustration of this approach is driving regulations. For example:

- The side of the road to drive on is typically strictly mandated.
- The type of vehicle that can be used, its color, power, model, can be chosen almost arbitrarily by the driver.

So the strict application of one rule (the side of the road to drive on), while technically a restriction on freedom, actually empowers drivers since it lets them know what side of the road everybody else will be driving on. In practice, there are some restrictions on the vehicle type as well, but they are generally broad enough that drivers feel free. So in reality, many of the strict driving rules free up capacity to choose and the ability to innovate in other areas.

In IT systems, much of the challenge is establishing where to be strict and where to allow freedom.

A way to express this as a policy dimension is: **Governance barriers -> Agreed interfaces**.

Working to establish which interfaces and core processes should be strict and adhered to by everybody benefits reliability by:

- Enforcing modeling discipline in key areas.
- Forcing communication about these interfaces and technology choices between groups.
- Creating a set of fixed points to measure drift, change, and degradation.

Surprisingly, this approach can also lead to increased productivity, such as:

- Interfaces at boundaries enable more radical change within domains. E.g., the adoption of a variety of implementation technologies.
- Teams waste less time interacting with other groups when potential problems are fixed ahead of time.

"The Developers' Exchange removes a lot of the friction of getting experts in the private sector to help solve the business problems of government. Any developer can download our source code and start working on it, without any access challenges."

Ian Bailey
Assistant Deputy Minister of Technology Services for the Office of the Chief Information Officer for the Government of British Columbia[11]

Self-service provisioning

Setting the goal of enabling IT capabilities everywhere, independent of the cloud location in which they might initially be needed (policy area two), can be thought of as a horizontal dimension to creating a uniform, consistent environment. However, there is also a vertical dimension to this notion – a view on how a capability is made available. A simple way to think about this approach is as a series of questions:

Figure 4. How self-service are your IT systems?

This spectrum for provisioning addresses two things:

1. The amount of time and effort required by team members to get set up with the resources they need.
2. The potential skill levels that team members require to access certain resources.

We express this continuum as: **Applications -> Self-service services**.

Many of the capabilities available in IT environments are installable applications that are different from test to production, are only available in certain places, and may require significant skills to set up correctly. The more self-service the applications and the application development tools are, the more productive the organization is likely to become. For example, organizations:

- Reduce the time required for developer teams to provision resources, evaluate options, and get productive.
- Allow individuals to contribute faster by removing the initial barrier of laborious system set up that often requires specialist knowledge.

There are also clear reliability benefits that come from using this approach:

- Fewer errors and debugging due to the automation of service delivery – especially during initial setup and configuration where errors can lead to undetected problems later, breaking up production execution.
- Higher service availability when provisioning if for many users across a larger, shared resource base than for an individual user.
- Better capacity and budget planning that leads to more reliable and efficient resource use.

11 *Red Hat customer case study, "Government of British Columbia improves services with open source." https://www.redhat.com/en/resources/government-of-british-columbia-case-study.*

Adopting a product mindset

An often mentioned and powerful principle in the design of microservices systems[12] and API-based architectures[13] is that of moving from a "project mindset" to a "product mindset." Specifically, in the high-pressure, continual operation of IT systems, it feels good to complete a list of projects. This approach makes sense for one-off issues or tasks, but it is not ideal when the deliverable is meant to be a durable system element on which other teams rely.

One way to think of this concept is as the difference between hunters and farmers. As a hunter, it feels good to capture the game needed for that day and get the projects completed. In contrast, farmers need a regular operation that produces results over time. Thinking about the whole application environment, real progress is the improvement of core assets, or the creation of new ones, which meaningfully upgrade and improve production for the long term.

We express this policy principle as: **Projects -> products**.

Thinking of IT systems as products causes us to consider ongoing production, the users, and what the users need. Then, if we're planning to make changes, how to best communicate them ahead of time? These are all things that would be applied to products used by customers. But this thinking is equally valuable—even critical—to apply to any system that affects any user group, including:

- Partners.
- Employees.
- Developers.
- Operations teams.

Without product thinking, system reuse is not likely to happen. But a commitment to product thinking can:

- Increase reliability by creating clearer ownership of reusable services and capabilities, create more predictable upgrade/downgrade/retirement cycles, and result in fewer surprises for operations and other teams.
- Increase productivity by strengthening the services and capabilities on which teams rely, reduce the need to rebuild or replicate locally, and allow internal services to potentially become valuable and tested external offerings in the future.

---

12 *Martin Fowler and James Lewis. "Microservices." Martin Fowler blog, March 25, 2014, https://martinfowler.com/articles/microservices.html.*

13 *Bortenschlager, Manfred, and Willmott, Steven. "The API Owner's Manual." https://www.redhat.com/en/resources/3scale-api-owners-manual-ebook.*

Tend toward antifragility

The "design for failure" principle has been popularized with cloud-native development. Container infrastructure has made it much easier to scale up and down in failure modes and the Netflix-inspired notion of chaos engineering[14] is becoming widely adopted—even in traditionally conservative sectors like banking.[15] These approaches are arguably an IT overview of a more general notion of "antifragility," coined by the writer Nicolas Nassim Taleb. In its simplest form, Taleb describes the concept: "Antifragility is beyond resilience or robustness. The resilient resists shocks and stays the same; the antifragile gets better."[16]

The core principle of antifragility is that systems get better as they are stressed. Ideally, principle would happen automatically (as in a system fixing itself), but diligent and persistent improvement by human intervention would count.

The more subtle, yet important, point behind antifragility is that human instinct—and the prevalent instinct in IT—is to focus on the protection of fragile systems over programs of improvement. We generally aim to prevent problems in the first place, rather than looking at how we would recover and grow from them when they inevitably happen. This tendency is one of the reasons chaos-engineering approaches are still rare in practice. Most IT systems are arrays of isolated, brittle systems, with protections in place to prevent random elements from causing a disturbance.

Seen from an IT perspective, antifragility really comprises two distinct functions:

1. Observability, or the ability to detect failures or certain bad behaviours.
2. Healing, or the ability to react appropriately to reduce damage and come back stronger.

Either or both of these stages may involve human intervention, although automation is becoming increasingly critical to react quickly.

The key mindset shift is to one that considers the range of failure modes that we can currently tolerate. And then, how do we detect them? How de we analyze data from failures? What systems should we put in place to grow as failures occur? How might we automate the process?

We express this policy principle as: **Cotton wool wrapped systems -> Antifragility**.

This axis is probably the most challenging for an IT organization to move along, and it is arguably the most valuable. There are huge gains to be made, even with gradual progress toward this goal. From a reliability perspective, looking at mechanisms to systematically improve beyond each failure gradually makes the system more robust. From a productivity perspective, less worry about breakage allows for faster experimentation cycles.

---

14 *Cameron, Lori M. "Chaos Engineering: It Sounds Scary, But Intentionally Harming Systems Can Find Bigger Bugs. How To Make The Cultural Shift, From Netflix Experts Who Do It." Computer, November 15, 2018. https://publications.computer.org/computer-magazine/2018/11/15/netlfix-chaos-engineering/.*

15 *Cowan, Paris. "NAB deploys Chaos Monkey to kill servers 24/7." IT News, April 9, 2014, https://www.itnews.com.au/news/nab-deploys-chaos-monkey-to-kill-servers-24-7-382285.*

16 *Taleb, Nassim Nicholas. Antifragile: Things That Gain from Disorder. United States: Random House, 2012.*

A final, important point is that while fragility and antifragility might be initially thought of in terms of systems and operations, these concepts are just as important for development processes, product development, and team collaboration. Any process that contributes to forward motion can benefit from some progress along the antifragility axis.

Automation as code

Automation has been noted as an important component in two previous policy areas, but it is so critical that we also wanted to focus on it explicitly.

Most IT systems today would not function if there wasn't already automation in place. But in many cases, automation designed in previous times of technical change can become part of the problem. Namely:

- Bespoke apps with only one function, written in a programming language that is no longer common in the organization—or the world.
- Complex, interwoven conjunctions of scripts that no one is willing to touch lest they fail.

What is the difference between good automation and bad automation? While Red Hat's The automated enterprise e-book describes automation best practices in detail, there is one general principle to understand: the difference between implicit and explicit automation.

Implicit automation is one-off code or scripting executing in place, as opposed to explicit automation that is recognized as code and is versioned, tested, updated, and managed explicitly as part of system configuration.

We express this policy transformation as: **Hand-crafted configuration -> Automation as code**.

Almost every organization has hand-crafted configurations in many parts of its systems. Over time, transforming these to be explicitly managed code creates a lot of value. For example:

- From a reliability perspective, it reduces the chance of error due to missed dependencies, speeds up change ability, and creates discipline in change processes.
- From a productivity perspective, it reduces the areas of systems that are poorly understood and reduces the amount of time spent on hard-to-diagnose failures.
- Automation as code also increases reusability and lowers the learning curve.

Overall, moving to automation as code is a prerequisite for true DevOps success because it separates configuration from code.

Security in depth

Perimeter security has been an insufficient approach to IT security for a long time, yet many organizations still rely on it either for the whole infrastructure, or within subgroups of datacenters. In hybrid cloud scenarios, not only are applications distributed between multiple datacenters and clouds, there are integrations and channels for data flow between some of these locations. This configuration means that compromised systems in one location could lead to unwanted access in others. Therefore, it's worth assessing security in at least three dimensions:

1. **Vertical.** From the operating system up through virtualizations, container management, and code execution runtimes, to the management of applications, good security requires consistent and up-to-date software in the entire stack.
2. **Horizontal.** With microservices patterns, APIs, and communication between applications, good security requires encryption, tracking, and access control for most, if not all, communication.
3. **Team and lifecycle.** Even if the right technologies and procedures have been researched and chosen, development and operations teams need to be able to apply those technologies and processes. As much as possible, these processes should be automated and enforced as part of the development cycle.

One shift to consider is whether or not to maintain separate security and development teams. Development teams nominally stick to security guidelines, but are typically only required to consider security issues in the short period before production code deploys. Alternatively, a collaborative, more distributed approach may be better, allowing:

- Development, operations, and security (DevSecOps) teams to collaborate much more closely.
- Security protections to be built-in from the start at the prototyping stage.
- As many core security procedures as possible to be automated and hard-wired into development and operations environments, making security decisions easier.

We express this policy principle as: **Perimeter security -> Pervasive security**.

Clearly, security has an important impact on the reliability axis of IT systems. Protecting processes and data integrity, security reduces downtime and the risk of catastrophic loss. But thinking through security models, expanding the responsibility across teams, and automating many of the processes (like preconfigured, permitted container images), means more preemptive planning takes place – for an even greater reduction of risk.

Security can be seen as a drag on productivity because it often involves multiple, complex development steps rather than an embedded password, direct database access, or some other shortcut. While this may be true in the short term, with enough automation and support for processes, security can improve productivity through reductions in future refactoring, and lost time responding to incidents. Having well-established and reusable security mechanisms ultimately makes developers' lives much easier. Therefore, reducing the complexity in setup and configuration is time well spent.

Bringing policy areas together

Not all policy areas will be of equal importance to every organization, and in some areas, different choices might make sense. However, policies broadly cover key areas where significant benefits can be unlocked when looking holistically at IT systems – see Table 1.

Table 1. Overview recommendations with side-by-side benefits

Recommendation	Reliability gains	Productivity gains
Decentralized authority	• Best practices are easier to apply • Control and supervision of key items • Focus on the most important issues	• Flexibility to make the best of local context and knowledge • A greater sense of autonomy and ownership
Universal capabilities	• Serve from multiple locations—built-in failover • Uniformity of execution • Speed of services	• Convention over configuration—it works the same way everywhere • Transferable skills
Constraints = freedom	• Modeling discipline • Explicit communication about interfaces • Fixed reference points to measure drift, change, and degradation	• Fixed interfaces at boundaries enable more radical change within domains—e.g., polyglot • Less time wasted across groups when interaction patterns are agreed upon in advance
Self-service	• Automation of service delivery and scaling up or down to avoid errors • Higher service availability • Better capacity planning and efficiency	• Speed of setup • Enable users with less technical specialty knowledge to use some services
Product mindset	• Greater and clearer ownership of services • Predictable cycles for upgrades, downgrades, retirement, etc.	• Stronger components to reuse • No need to rebuild locally • Internal and external use of the same product is more efficient

Recommendation	Reliability gains	Productivity gains
Tend toward antifragility	• Big improvements in reliability—learn from every failure and fix • Grow tolerance of breakage range • Decouple dependencies to avoid cascading failures	• Less worry about breakage • Faster experimentation cycles
Automation as code	• Reduce error by removing human steps required • Speed up change when fixes need to be deployed • Create discipline in designing change processes	• Remove or speed up mundane tasks • Less failure means more time for innovation
Security in depth	• Fewer security incidents improve uptime • Preemptive planning for issues at multiple levels gets more team members thinking about security	• Include security into some systems from the outset • Shared responsibility may slow things down initially, but eventually speeds up the whole

From policy to actions—some advice

Once policies and guide rails are established, the next step is to determine actions that move the organization and deployed systems forward to the desired state. These actions can be directed toward the general outcomes planned, reinforcement of patterns, or specific projects. The next two sections offer considerations for this critical implementation phase, but we'll start with some general advice:

- **Focus on environment-wide evolution, not revolution.** Most organizations already have a complex set of initiatives, projects, and activities in progress for different parts of the business, so adding more could be daunting. The most valuable contribution of an environment-wide application strategy is to assess change in the context of the health of the whole organization. Evolution across the whole supports this goal more than revolutionary change in a few units. This approach doesn't mean hero/heroine teams don't have their place. In fact, those achievements should be celebrated examples of improvement of the whole.
- **Think about systems and metrics just as much as goals.** Setting goals for improvements is often relatively straightforward. It's easy to say "we need to get better at X." But goals of this nature, even with actions attached, may only attract a short-term burst of activity. A more effective approach is to select one or more metrics that capture the status of the environment in this dimension and work toward a set of behaviors and habits that measures and moves the metric forward.

Table 2. Sample metrics and processes for policy areas discussed in Section two

Recommendation	Sample metric	Type of process
Decentralized authority	Decision time for changes or exception scored by the size of the issue	• Regular cadence of global and group planning processes to set policies • Change request or exception processes between groups
Access everywhere	Availability of services and speed or service-level agreements (SLAs) per region, datacenter, or cloud	• Gradual prioritized roll out of services across properties • Assessment and continuous testing of functionality parity
Constraints = freedom	• Adoption of design frameworks, patterns. or guidelines • Compliance of solutions • Number of exception requests	• Organization-wide sharing of interfaces, products, and internal services • Regular cadence recognition of the adoption of central guidelines
Self-service	Speed, or the number of steps to access key services or products with a metric such as, "Time to First Hello World" for an API being used, or "Time to first third-party user" for an API being used	Regular cadence cycle of updated lists of available services and measure times required to access them by their users
Product mindset	The number of groups, services, and products using a particular product or service	Regular cadence on the release of products and services, feedback mechanisms, communication means to update on changes, sharing usage metrics
Tend toward antifragility	Measure of time to heal after failures, as well as the (hopefully decreasing) trend on recurrence of similar failure	• Regular chaos tests inducing simulated or real failures across system to detect fragility • Gradual widening of the envelope of the severity of system shocks induced

Recommendation	Sample metric	Type of process
Automation everywhere	Measure the number of manual deployment or configurations or other steps—or total end-to-end time taken for full redeploys	Regular review of automation metrics and ongoing program to reduce the largest pain points
Security in depth	• Number of security incidents detected and caught • Number of components adopting security guidelines and monitored	• Regular penetration and other security testing to stress system • Post action reviews

Part three: Architecture for success

The strategy principles in the previous sections are abstract so that they might be applied to a variety of systems. The next logical questions are:

- What do these systems look like?
- What are the components to consider, and how are they organized?

There is no single architecture that works for all cases. Architectural choices will always be unique to an organization, often representing core strengths to build upon. However, it is useful to provide a map of the typical elements of an application environment and how they connect when cloud-native and hybrid cloud technologies are applied.

In this section, we'll provide an overall picture of what this means, and drill down into a number of areas that have become critical to modern IT.

The big picture

Borrowing from our earlier definition of an application environment as the set of capabilities for application delivery and application development across the entire organization, a high-level view of a large, organization-wide application environment is illustrated in Figure 5.

"The moment we have an idea, we can start building the product. This agility is something we have never experienced before."

Tobias Mohr
Head of Technology and Infrastructure, Aviatar, Lufthansa Technik[17]

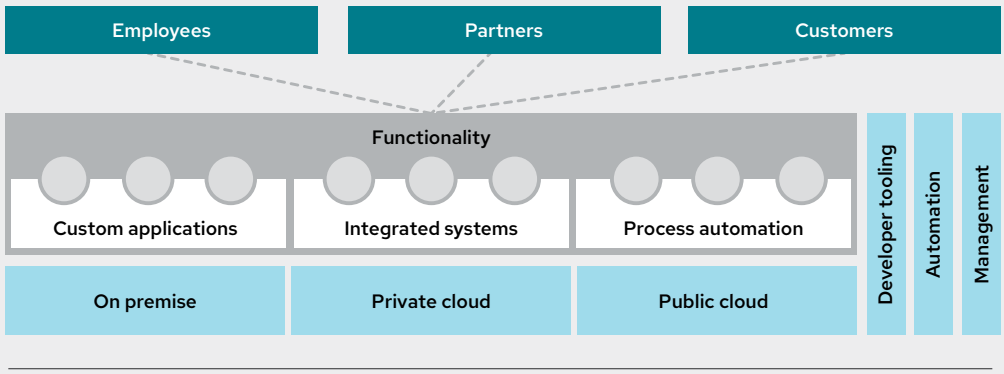


Figure 5. High-level IT application environment

In this section, we focus on the components of an application environment and how they are connected. Each of these layers can be considered perpendicular to the strategy principles discussed earlier in Part two. The strategy principles should hold across these layers and inform their design. The division is as follows:

1. **Platforms and application delivery.** The infrastructure that hosts the code processes that instantiate the applications which deliver value at the operating system and datacenter level.
2. **Applications from custom code.** Support for the execution of custom-code-based applications that are innovative in functionality and separates an organization from its competitors.
3. **Applications from integration.** Support for the communication between applications that creates the functionality users experience directly.
4. **Applications from process automation.** Support for applications that involve not only executing code, but also processes that engage humans (workflows), and logical rules contributed by non-developer experts in the organization.
5. **Developer tooling, DevOps, and management.** Capabilities that wrap around the application environment to keep team members productive and systems functioning.

Each area is addressed in its own section to follow. In addition, a crucial success factor in implementing these strategies and architectures is addressing human processes and architectural patterns. Team dynamics, behaviors, and culture that maintain and evolve the system are all important factors for success. They also capture the organizational memory of the approaches that solve problems. These factors are addressed in Part four, and a detailed analysis can be found in *The Open Organization*.[18]

17 *Red Hat success story, "Lufthansa Technik builds cloud platform to optimize airline operations." https://www.redhat.com/en/success-stories/lufthansa-technik*

18 *Whitehurst, Jim, The Open Organization, Boston, Massachusetts: Harvard Business Review Press, 2015.*

"The place has clearly been terraformed within an inch of its life; there was just no other way continents ended up perfectly square."

Rachel Bach
Author, *Heaven's Queen*[19]

There are other ways to segment IT infrastructure, depending on the focus, for example, mobile application support or Internet of Things (IoT). However, the model here is deliberately structured to illustrate the importance of the whole set of connected elements that make up an IT environment. In general, most IT strategy advice is either focused only on custom code development, ignoring that much of IT is about integration and processes, or assumes the adoption of a particular technology stack.

The application environments of most large organizations are vast and diverse, naturally spanning many forms of development. Typical organizations operate on the basis of thousands of individual applications, a wide range of SaaS services, commercial off-the-shelf solutions, and other systems. The architecture layers discussed provide a general overview of how to connect some of these systems. The examples refer to Red Hat technologies, where relevant, but there are certainly other vendor solutions available. Any sufficiently complex system will need to rely on technology from many sources. As such, the technology-specific items listed are for illustrative purposes only.

The five levels of IT terraforming hierarchy:

1) Platforms -> soil and nutrients

2) Applications -> atomic life, plants and small animals

3) Integration/messaging -> communities

4) Processes -> civilization

5) Developer tools / DevOps and management -> ecosystem processes and checks and balances

Platform and delivery

Improvements in platform and application delivery technology have arguably been the most obvious drivers of cloud-native and hybrid cloud capabilities to date. The platform layers provide the substrate on which applications execute, integrate, and deliver value. The shift from the operating system to virtualization to containerized deployments now allows for a broad and deep spectrum of different kinds of deployment and automation. This shift in technology also furthers the idea that the exact same execution environment can be available to developers in an organization, no matter where they deploy: in the datacenter or on any one of a number of public or private clouds.

In hybrid cloud environments with multiple on-premise locations, there is now an even greater premium on:

- **Consistency of experience across locations** ensures the same operating system versions, patch levels, and other system elements are available in as many of the environments as possible.
- **Unified management** across on-premise, virtualized, private, and public cloud systems, making it easy to track the entire IT infrastructure in the same way.
- **Control over security and compliance** continues to be critical even as different cloud and datacenter deployments proliferate.
- **Increased stability, agility, and availability.** Often, the goal of infrastructure expansion is to add capacity for failover and redundancy. However, it is important to take a consistent approach to these deployments or outlier environments could hurt reliability.

An in-depth look at Red Hat's operating system and private cloud approach to hybrid cloud, with many customer examples, can be found in Red Hat's hybrid cloud strategy e-book. However, the last few years have revealed an addition to creating a consistent platform across multiple cloud environments: the use of containers and container platforms.

19 *Bach, Rachel. Heaven's Queen (Paradox Book 3). Orbit, April 22, 2014.*

Containers have become the deployment infrastructure of choice in modern IT. These constructs package and isolate applications with their entire runtime environment – all of the files necessary to run. Rapid deployment and easy management via a container platform means that instances of applications can be deployed, updated, and scaled anywhere the container platform is available. Containers and container platforms truly bring the cloud-native computing vision to life on top of the hybrid cloud.

In a hybrid cloud environment, using a container platform makes it possible to create a truly identical computer environment across all datacenter and cloud environments by:

- Providing identical compute infrastructure in each location.
- Enabling a packaged container to run anywhere within the hybrid cloud environment without modification.
- Enabling operations teams to see workloads across their various container clusters.
- Enabling the scaling up and down of resources in each location.

The container platform creates an abstraction layer over the underlying resources, making them reusable and consistent everywhere. The Kubernetes-based Red Hat OpenShift® Container Platform provides all of these features and more. The mix of locations in any customer's hybrid cloud can now include hosted offerings on each of the three major public clouds – Amazon Web Services (AWS), Google Cloud Platform, and Microsoft Azure. More information on the use of containers for hybrid cloud can be found in the Red Hat cloud containers e-book.

Three platform strategy considerations are:

- **Making the platform as self-service as possible** to enable different teams to access the capacity and services they need while having a tight and consistent security model. Allowing development teams to self-provision virtual resources is crucial for speed. However, self-provisioning needs backed-in controls to provide system security.
- **The reliability of the entire platform infrastructure across all locations.** This consideration involves looking not only at individual systems or datacenters, but also at their interdependencies and change policies for when failures occur: Are systems simply "restored" or are they changed to prevent future failures?
- **Automation is key in complex hybrid cloud environments.** Building out automation is critical to keeping hybrid cloud management feasible. At the same time, as discussed in Part three, it is important to consider automation as code that must be managed. Otherwise, automation itself becomes a legacy problem.

Applications from custom code

Application code does not execute by itself. Executing source code is supported by substrates that have slowly evolved from early application servers to a mix of modern application servers, lightweight language runtimes, and a combination with container management.

"We can deploy new applications quickly, but the real beauty is the flexibility: some existing applications grow, others shrink. We can make those changes automatically with one click, in a minute instead of a day."

Ivan Torreblanca
CIO, Leshop.ch[20]

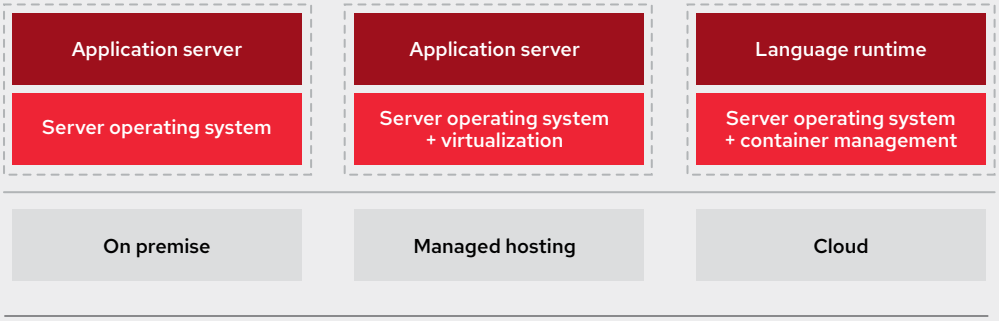


Figure 6. Execution of custom code in different contexts throughout the enterprise

As cloud-native technologies take hold, container infrastructure, in combination with language runtimes, provides the most flexible combination for innovation. As organizations adopt container technologies, this combination is likely to become the dominant deployment pattern with much of the previous application servers' orchestration capability being transferred to the container management layer.

For most organizations, this transition will take time, and it will likely have a mix of application hosting scenarios, including all of the approaches shown in Figure 6.

The following paradigms have emerged in application development and are being considered by the majority of large organizations:

- While Java and Javascript remain dominant in the enterprise for custom code implementations, the number of languages in use continues to grow. Cloud Foundry's recent report covers 25 languages in use, with many in the low single-digit percentages of enterprise respondents.[21]
- Tight integration and the inclusion of ancillary services for messaging and in-memory data grid are becoming important services in the environment in order for custom code projects to be efficient.
- Reactive programming is gaining significant traction as the need for rapid processing custom applications increases. Toolkits, such as Eclipse Vert.x, have been seeing swift adoption.
- Function-as-a-Service (FaaS), first popularized through AWS's serverless Lambda approach and now having a more generalized computer approach, allows programmers not to worry about pre-provisioning compute capacity. Instead, individual elements of code (functions) can be deployed and executed simply when the right events occur. FaaS solutions are now available on the major public clouds and can be provisioned (typically on a container platform) by on-premise operations teams for their own teams of developers.

These trends add significant new capabilities to enterprise IT for the development of custom code applications. Carefully selecting which technologies to adopt is important because the custom code systems, which implement the bulk of the differentiating functionality, make an organization uniquely competitive in the market.

20 *Red Hat success story, "LeShop.ch supports culture of innovation with agile, scalable solution." https://www.redhat.com/en/success-stories/leshop.ch*

21 *Cloud Foundry, "These Are the Top Languages for Enterprise Application Development." August 2018. www.cloudfoundry.org/wp-content/uploads/Developer-Language-Report_FINAL.pdf.*

Red Hat's application development portfolio is broad and deep and covers most of the areas already identified in this section. Some recent innovations include:

- A broad expansion of the enterprise-grade runtimes available to support custom code. Red Hat's all-inclusive runtimes bundle now include JBoss® Enterprise Application Platform, Node.js, MicroProfile, Spring Boot, and others. This bundle provides a broad range of flexibility so developers can choose the tools they prefer, while CIOs can continue to minimize risk across the whole spectrum of an organization's custom code.
- The inclusion of complementary services for messaging and in-memory data grid, with Red Hat AMQ broker and Red Hat Data Grid.
- Tight integration with Kubernetes and Red Hat OpenShift Container Platform, enabling a clean separation of the lower-level deployment capabilities traditionally handled by an application server. These capabilities can now be handled in the container platform through the code-centric runtime support services in the runtimes packages.
- Support for Kubernetes-native execution mechanisms that enable FaaS wherever a Kubernetes container platform is deployed. Known as Knative, this collaboration with Google, Pivotal, and others, provides the substrate for messaging, eventing, and execution of a serverless approach to work anywhere.
- Lastly, in cloud-native environments, execution speed is critical. Applications that teams want to use in cloud, container, or even serverless environments should start up and shut down in micro-seconds: scaling to zero when they are not needed, but being available nearly instantly when called upon. Red Hat's Quarkus technology now provides this speed and agility for Java virtual machine (JVM)-based languages.

Key considerations of application runtime strategy are:

- Autonomy with consistency: enabling the use of different languages and approaches in throughout the organization.
- Tight integration with DevOps and developer tools is important for developer productivity. Automation of standard processes, checks, and balances allow more code to be written, tested, and deployed efficiently.
- Establishing a comprehensive security strategy ensures that from the operating system through virtualization and from the container layer through to the code runtime, all systems are comprehen-sively updated by automated rollouts from trusted vendors.

Applications from integration

Since IT systems were first connected to networks and operating systems first allowed interprocess communication, integration capabilities have been necessary. The number of ways software systems can be integrated has ballooned. Integration has become one of the most critical parts of the IT stack.

With application environments now spanning multiple datacenters and clouds, integration is critical, but they also need to adapt to modern requirements. While the traditional patterns for integration, such as enterprise service bus (ESB) deployments, have brought organizations a long way, they are far from sufficient for hybrid cloud challenges. In particular, there are a number of challenges integra-tion technology needs to meet:

"Our Red Hat operating environment frees us to focus far less on our infrastructure and spend more time innovating and delivering new products that improve our customer experience."

Michael Catuara
Director of Distributed Systems, TransUnion[22]

- **Seamless operation across multiple locations.** This operation means that messaging, API management, transformation, data mapping, and other capabilities should all be available in multiple physical locations. The capabilities also need to work together across locations. For example, messaging solutions often need to be used across multiple physical locations, not just within a location. IDC terms this capability "portability" in a recent report.[23]
- **Integration as code.** Integrations between systems perform critical transformation and data mapping in real time. This capability requires code changes to the software endpoints, delivering data that can imply synchronized changes to the integration. Integrations should be considered code just like applications so that they may be versioned in the same way.
- **Extreme scale.** While integration technology started in the datacenter connecting primarily back-office applications, in many cases customer-facing applications need access to those same back-office systems. Meanwhile, traffic volumes have increased significantly. Integrations, therefore, need to scale rapidly and cost effectively for the most heavily used data and transaction systems.
- **Serving more stakeholders.** The number of systems has grown rapidly and IT has infiltrated almost every business function. The number and type of integrations required in an organization has dramatically increased. Supporting integration activities is becoming increasingly critical not only for development and operations teams, but also less technically trained individuals in the line of business areas, known as citizen integrators.
- **The increasing importance of data.** In line with their use in customer-facing applications, integrations are also front and center in the trend toward large-scale, real-time data capture and analysis. Moreover, the proliferation of IoT devices requires new ways to manage the vast amount of information they generate. Data needs to be processed, stored, and often replicated accurately for modern systems to operate effectively.

To cope with these changes, integration technologies themselves are evolving. As they combine with cloud-native capabilities, integration technologies in the Red Hat portfolio:

- Can be deployed in a fully distributed manner, with individual integrations wherever they are needed.
- Are container-native, which permits automated capacity scaling.
- Can be used on demand with instant start up and shut down via container-based Knative technology.
- All integrations are manipulatable as code, which means normal DevOps practices can be applied and are accessible via graphical drag and drop interface. This capability enables highly skilled and less-skilled individuals to collaborate on the same integrations.
- Includes an even broader range of messaging capabilities, from traditional AMQ-style messaging to Apache Kafka.
- Messaging, API, and integration technologies can be deployed across different clouds and provide the application-level communication needed for end-user applications to operate.

---

22 *Red Hat success story, "TransUnion modernizes IT, delivers a better customer experience." www.redhat.com/en/success-stories/transunion*

23 *Fleming, Maureen. "Worldwide Integration and API Management Software Forecast, 2019–2023," IDC, June 2019, https://www.idc.com/getdoc.jsp?containerId=US45126319.*

"Our business grows dramatically between Thanksgiving and the end of December. There's more online shopping happening, but we're also seeing returns expand our busiest time. Using OpenShift, we can scale flexibly during those particular peak times. We even have the potential, if needed, to scale to public cloud."

Carla Maier
Senior Manager, Cloud Platforms and Technology, UPS[24]

The most important integration considerations for organizations are:

- Mapping out a strategy for which products and services become available and how they will be accessed by other software teams. Often APIs are an important part of this broader adoption.
- Determining a strategy for who owns integrations between these systems. Distributing this workload increases the autonomy of individual groups and reduces the workload on centralized Integration Competency Center (ICC) teams.
- Addressing strategic questions, such as setting best practices for changed-data capture, virtualization of federated data, messaging, and data transformation across datacenter and cloud integrations. How will these capabilities connect with the platform layer and provide seamless service everywhere?

A deeper dive into integration strategy in cloud-native and hybrid cloud environments can be found in Red Hat's Agile integration e-book.

Applications from process automation

Software touches nearly every process an organization executes. Many of these processes involve people (employees, partners, and customers) and also complex decisions based on data and states.

The third type of application that is radically changing in the hybrid cloud era is process-driven applications.

There is a clear, growing need for both business process management (BPM) applications and automated intelligent decision making, covering everything from rule-based systems to robotic process automation. As IT spreads infrastructure through the hybrid cloud and influence throughout their organization, the following trends are emerging for process-driven applications:

- **The adoption of microservices patterns, APIs, and integrations** creates an opportunity to connect processes with new systems throughout the enterprise—and build richer workflows and apply automation to data sets or systems that were not previously accessible.
- **Solutions using configuration, business-level logic, and rules are becoming increasingly important.** Although custom applications deliver much of the novel functionality available in the environment, the ability to call that custom logic in ways that reflect changing business needs is critical. Configurable rules are capable of executing actions as FaaS invocations in the cloud and containers.
- **An increase in process distribution** as the number of datacenter and cloud locations in an organization grows, and processes need to span these locations, requiring a cloud-native approach to process management. This approach is not simply deployed in one location, but it can be configured to run wherever it is needed and create consistent flows across all locations.
- **The rise of BizDevOps,** which aims to combine business needs directly with agile application development. In particular, toolchains are becoming more integrated to connect code concept to cash.

24 *Red Hat case study, "UPS streamlines package tracking and delivery with DevOps and Red Hat." www.redhat.com/en/resources/ups-customer-case-study*

"We have built up hundreds of interfaces to multiple EMR, lab and other systems across the enterprise. It's a huge challenge. The health information exchange team has been at it for the last five to six years. But we are in a good position where most of our data is integrated and easily available in a single place. That data can be mined for analytic insights, or leveraged by a decision engine to make recommendations."

Vipul Kashyap
Director of Clinical Information Systems and Enterprise Information Architect, Northwell Health[25]

- **Scalability.** Process-driven applications, particularly in organizations with large numbers of employees or customers, now underpin some of the most highly used IT services. As a result, it has become necessary to scale up and down usage of key processes on demand to avoid failures and bottlenecks.
- **The use of serverless and FaaS platforms** utilized by business rules and business process management systems to help with scalability, performance, and cost savings. Decisions and their actions can be executed as FaaS. Also, steps in business processes can also be executed as FaaS.
- **The confluence of IT integration between back-office systems and customer-facing systems, and the resulting growth in data,** has created an enormous opportunity for organizations to begin deploying intelligent applications for their clients. Making rule-based decisions, splicing together human and automated decision making, and doing this with speed and scale are all key to taking advantage of these opportunities.

Each of these trends means that a modern wave of process-driven applications is now a crucial component of IT strategy.

As with integration technologies, cloud-native development is also enabling major changes in process automation technologies. In the Red Hat portfolio, it means:

- **Native deployment on containers.** Tight integration with containers and container orchestration technology is a core part of Red Hat's process automation platforms. Process-driven applications benefit from the scalability, resilience, and location independence provided by a cloud-native architecture.
- **Serverless and FaaS.** Red Hat's process automation solution, Kogito, extends cloud-native capabilities by supporting serverless and FaaS models for process-driven applications. Kogito is built from the ground up with Quarkus to provide the rapid startup, scaling, and memory efficiency needed for on-demand services.
- **Cognitive computing.** Cloud-native architecture provides the scale necessary to support resource-intensive artificial intelligence (AI) and machine learning (ML) workloads, which can extend decision services to cover a wider range of scenarios. By incorporating predictive models into user-written Decision Model and Notation (DMN) decision models, Red Hat Decision Manager enables organizations to increase automation and reduce exceptions requiring human input.
- **Intelligent management.** Cloud-native development reduces operations workload through Kubernetes Operators, which automates application life-cycle management and fault response. Red Hat's process automation solutions include Operators for process-driven applications to easily manage distributed hybrid cloud workloads.

The strategic decisions relevant to this area include:

- **Enabling increasing numbers of employees, partners, and customers to participate in the creation and configuration of process-driven applications.** This type of application creates an opportunity for decentralized authority where a safe, structured environment can be provided by technical specialists. Domain experts can then use this to control the applications for their own needs.

---

25 *RTInsights analyst paper, "Building a clinical decision engine platform with Red Hat Decision Manager," 2018. www.redhat.com/en/resources/building-a-clinical-decision-engine-rtinsights-article.*

"OpenShift and OpenStack allow our teams to develop an idea quickly, eliminating distractions from innovation and discovery. They don't need to worry about whether they need a small or big server to run their apps."

José María Ruesta
Global Head of Infrastructure, Service and Open Systems, BBVA[26]

- **Determining protocols and procedures, roles and responsibilities** for collaboration between line-of-business and IT when automating manual business processes.
- **Deciding on approaches to use FaaS and the cloud for decision and business process management (BPM).** For example, whether to use BPM-as-a-Service on the cloud, on-premise, or a combination of both.
- **Providing process management on-demand as a service may bring the greatest reward,** allowing teams that previously did not have process specialists to begin capturing and instrumenting their own workflows.
- **Moving toward antifragility.** By replacing custom-engineered, compiled code for processes with configurable rules and standard formats, complex processes can be better validated and change does not need to involve code deployment. This approach reduces the risk of system breakage and helps more team members understand the dependencies inherent within important processes.

Developer tooling, devops, and application management

DevOps and developer tooling represent the foundation of cloud-native technology. In essence, many cloud-native technologies have evolved from the core need for infrastructure to support developers more efficiently and effectively. The Cloud Native Computing Foundation's (CNCF) definition of cloud-native technology starts with:[27]

"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds."

CNCF covers some examples and then highlights:

"These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil."

This combination of resilience, manageability, and observability, (from the operations side) and engineering impact (from the developer and engineering side) has fostered a number of technologies that have come together under the cloud-native banner.

These attributes are what makes cloud-native technology a compelling technology set used to address many of the challenges that emerge as organizations move along their trajectory of adopting multiple clouds and datacenters as their infrastructure footprint.

DevOps practices, application services, middleware, developer tools and management, as well as new cloud-native variants, all comprise and support the IT environment of an organization. They work together to create the organization's application environment. In turn, that application environment supports the functionality that enables employees, partners, and customers to derive the value they need.

---

26 *Red Hat case study, "BBVA transforms customer experience with cloud-native digital platform." https://www.redhat.com/en/resources/bbva-customer-case-study*

27 *From the Cloud Native Computing Foundation (CNCF) official definition of cloud native v1.0 https://github.com/cncf/toc/blob/master/DEFINITION.md*

Much has been written about the successful application of DevOps techniques in organizations – the Red Hat DevOps resources page is a great starting point. Also, Red Hat Open Innovation Labs and other services teams provide programs to help get started. Rather than repeat this general guidance, this section focuses on a number of trends accelerated by hybrid cloud adoption:

- The notion of infrastructure as code, which comes from cloud-native DevOps, is becoming critical as infrastructure and applications span multiple datacenters and clouds. Without the ability to fully automate deployments in multiple locations, hybrid cloud environments become increasingly hard to manage.
- This need for automation extends to application code as well. Since applications are now tightly woven integrations across numerous services and APIs in multiple locations, deployments need to be carefully orchestrated to avoid downtime.
- Almost all IT environments are mixed platform environments that can include bare-metal deployed machines running a variety of operating systems, virtualized server real estate, and private cloud, public cloud, and containers. The mixture of these different platform levels means that any holistic approach needs to connect management practices across platforms.
- Development tools need to change and update with infrastructure, providing live testing and other cloud-native features. Cloud-integrated development environments, in which code lives entirely in the browser, are gaining traction to meet this need.
- Observability and tracing becomes harder and more crucial in hybrid cloud environments. Code execution often touches multiple locations and must be tracked across each invocation. Failures in disparate parts of the system need to be correlated to determine what went wrong.

Each of these trends raises new challenges, but today's tools are evolving rapidly to cope with these struggles. Each challenge results from an increasing decentralization of infrastructure across locations, application types, and technologies. Part of the response is to standardize elements of deployments. A second part is to deploy tools and architecture patterns that explicitly solve for the distributed nature of the system. Both are needed for success.

Some of the developments in Red Hat's solution stack include:

- Multicluster management in Red Hat OpenShift, as well as a variety of tools for integrating continuous integration and continuous delivery (CI/CD) pipelines with container deployments that create a powerful hybrid cloud environment.
- Red Hat Developer's CodeReady workspaces provide a completely web browser-based integrated development environment (IDE) optimized for container-deployed code. Features include container-based workspaces, prebuilt or custom stacks, centralized management of developer configurations, and much more.
- Red Hat Ansible® Automation Platform simplifies and adds security to management across hybrid IT environments, targeting the hybrid cloud with workflow features and security standards.[28]
- Red Hat's predictive monitoring solution, Red Hat Insights, is now included with all Red Hat Enterprise Linux® subscriptions. The platform provides integrated security, performance, availability, and stability across a variety of Red Hat platforms to increase visibility in the hybrid cloud.

28 *Red Hat press release."Red Hat Unifies Automation Across Hybrid Cloud Management with Latest Version of Red Hat Ansible Tower," January 9, 2019, https://www.ansible.com/press-center/press-releases/red-hat-ansible-tower-3-4.*

From a long-term planning perspective, the following strategic areas are key for developer tooling, DevOps, and application management:

- **Universal capabilities** ensure that the same underlying infrastructure, services, and automation is available in each location and for the application delivery infrastructure (platform and automation) and the applications themselves. Often deployments need to run the latest version of each app in multiple locations.
- **Developer tooling approach.** For example, a browser-based IDE running on containers may be the best option for providing a consistent developer experience across on-premise and cloud environments.
- **A product mindset among development and operations teams.** Creating applications and services for reuse is hard work. However, it is harder to build trust in product and services operated by others. A product mindset requires teams to think long term about the systems they provide. This impacts their planning process and will need to be captured in the automation, daily habits, and practices that go with updating, deploying, and retiring code.
- **Automation.** Without careful management of the processes and automations themselves, a successful DevOps implementation quickly becomes a rigid system with diminishing returns.

Part four: The application environment journey

The Red Hat IT team assessed the value and readiness for change of each of the applications making up Red Hat's business (see Figure 7). The team applied value criteria such as rate of change of the application and its lifespan, as well as readiness criteria including business criticality, business support, and architectural gaps.

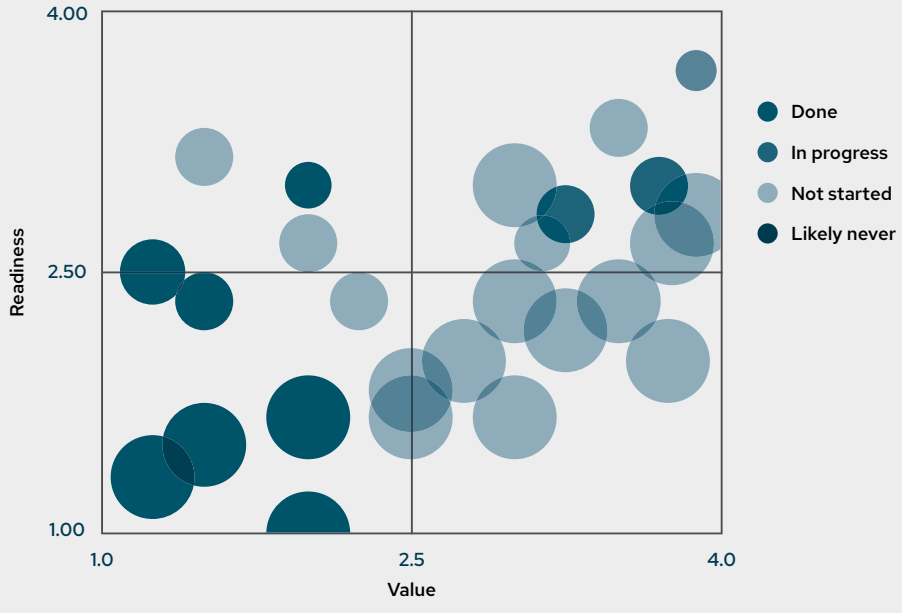


Figure 7. Application assessments for modernization (courtesy of Red Hat IT)

Except for newly established startups, there are very few situations where IT strategy is purely about the new. Typically, IT strategy is a study in how to evolve the entire functioning system for reliability and productivity improvement. Individual bursts of fast productivity may sometimes work, but without the general use of best practices, these instances end up draining resources.

This section covers the journey of cloud-native and hybrid cloud application environments by sequencing the strategy areas from Part three into Figure 8.

Figure 8. Linear representation of the evolution of resilience and productivity in application environments

It is important to note that these are not technology adoption stages. There may be technologies that are never adopted in some (or many) parts of a particular organization. The stages reflect policies, capabilities, and decision making.

It is also unrealistic to expect large, complex IT organizations to advance all the systems, processes, and strategic directions at once or in lock-step across the organization.

Visualizing these elements as a sequence is designed to highlight the logical sequencing of improvements and, more importantly, emphasize the whole application environment as a single entity. Improvements can be made in any unit, at any stage, but it is useful to consider how those changes affect the whole environment: Will they spread? Or will they stay segmented and disappear?

Fragmented bureaucracy

Fragmented bureaucracy might seem like an unkind description of an organization. However, given the complexity of today's IT systems, this is where many IT organizations operate. The rules, processes, and technology choices of previous builder generations are what made the systems and the organization. These decisions are reflected in the IT organization. It is also unlikely that any set of changes, no matter how great or how persistent, will make the organization and its systems perfect. Any moment of perfection would become obsolete by the next change. As Gartner[29] describes, managing technical debt is not easy. In fact, IT organizations aren't just implementing an application – they are always implementing a complex ecosystem with a potential multi-decade lifespan. Functionality is transient while structure is permanent.

29 *Kyte, Andy. "Gartner Keynote: Everything You Always Wanted to Know About Technical Debt (but Were Afraid to Ask)." Gartner Application Strategies & Solutions Summit, November 27-29, 2018. https://emtevr.gcom.cloud/events/apn32/sessions/apn32%20-%20k4%20-%20gartner%20keynote%20everything%20you%20always%20wanted%20to%20k%20-%20373224_47073.pdf.*

It is important to not just get away from the label, but to move the organization and structures forward to improve. Bureaucracies are required structures to operate such large, complex organizations of people and technical systems—some fragmentation will always be present and even necessary. The questions to consider are:

- How good is the current structure and the application environment that it creates for the organization?
- What are the most important directions and mechanisms of change to keep the whole improving?

Steps along the cloud-native and hybrid cloud path

For many IT organizations, using multiple datacenters and multiple public cloud locations, often from more than one cloud provider, is now a daily reality. From a technology perspective, there are a variety of strategies to pursue that help make such environments more manageable.

Improving the application environment that powers your organization might require a range of technology, but above all, it requires making strategic decisions about prioritizing areas of change. Seen as a progression, the following four areas look like steps:

- **Product mindset.** Adopting a product mindset and cultivating it for any system that is intended for reuse helps teams act more autonomously. It is also a prerequisite for teams in the organization to rely on and trust systems run by others. In fact, this focus is also a characteristic of a microservice architecture that advocates that a team should own a product over its lifetime, from development to production maintenance, resulting in a closer linkage to business capabilities and how they can be enhanced. A product mindset helps guide the discourse on which interfaces and processes are truly important. Where are the domain boundaries and which APIs and functionality sets need solid commitments behind them?
- **Services everywhere.** Emphasize the need to think beyond initial deployments and plan for seamless functionality to be available in multiple locations. There can be costs to this planning (or rollout) but they are usually lower than dealing with wildly different solutions to the same problem. This area also encompasses the push for self service, where possible so developers, and potentially users, can self-provision resources. There are significant efficiency and productivity gains in running an infrastructure in this way. These gains will grow in importance as container and serverless type technologies become more prevalent.
- **Automation as code.** Raising the level of automation across the IT environment provides a higher return on investment, but only if this automation is explicitly maintained in the same way code and applications are maintained. Without explicit management, today's automation may become tomorrow's awkward legacy. A similar mindset shift applies to integrations, decision-making rules, and process automation code. These generate or underpin application functionality. In many cases, more integration code is present in a system than custom logic. Explicitly considering these systems as code makes it easier to manage deployments across systems.
- **Antifragility.** The move toward antifragility represents the presence of strong processes (automated or human) that react to failures and improve the system beyond where it was before. Few organizations have adopted extreme, Netflix-like chaos engineering at any significant scale, so progress towards antifragility is a valuable goal. Similarly, including security in this, and all previous, steps represents a significant leap towards proactivity.

Feedback loops and culture

Adopting new technology and processes always comes with pain and uncertainty. In many cases, it is not advisable to change something that is already working. One of the lessons the Red Hat IT team learned is that it is critical to listen to both skeptics and champions regarding change.

Often both camps with strong opinions have valid points to make. "Why are we implementing B when A already works?" While phrases like this indicate some sentimental attachment to A, it may also indicate that B is unlikely to address the underlying problem.

A useful exercise carried out by the Red Hat IT team was mapping applications to capabilities and groups according to their strategic nature, as shown in Figure 9.

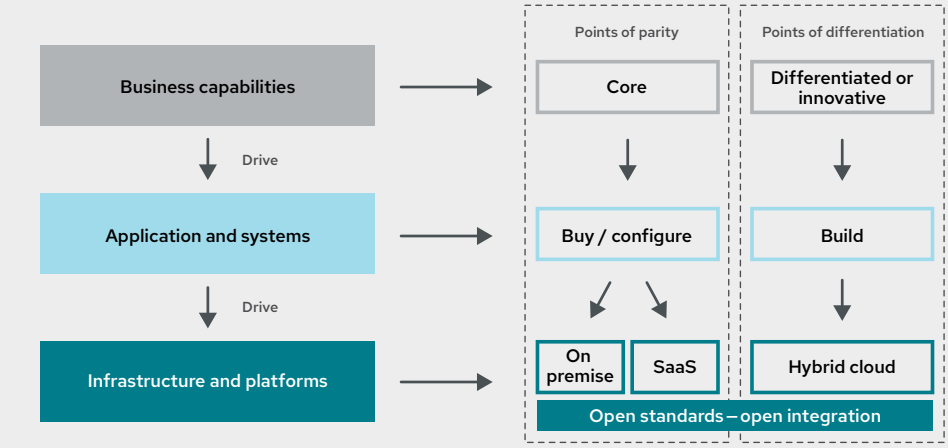


Figure 9. Classifying applications into points of parity and points of differentiation helps teams determine where energy is best spent

The team also determined which applications to migrate and when to build confidence in new processes. Each application and process change may be different, but the team gains trust in decision making and execution along the way.

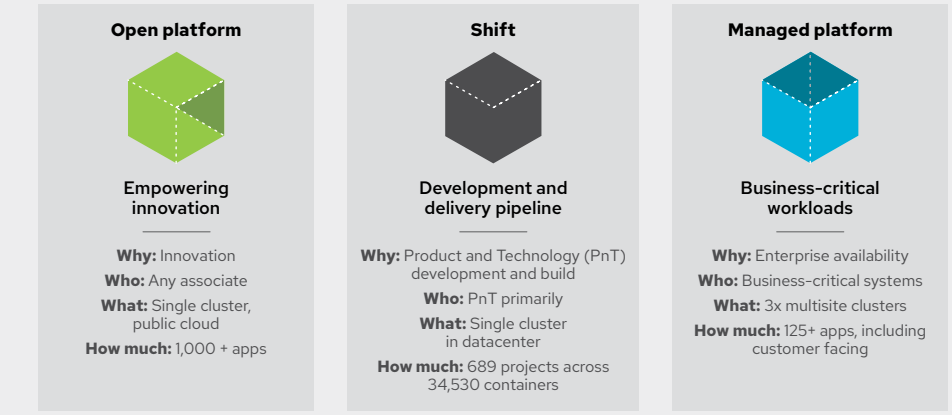


Figure 10. Parallel platforms used in Red Hat IT's adoption of hybrid cloud

A final strategy being employed at Red Hat is the internal rollout of platforms for different purposes, not just a single platform to serve Red Hat IT. In Red Hat IT's adoption of hybrid cloud, three parallel platforms were rolled out. In addition to the business-critical Managed Platform, the UpShift platform supports Red Hat Product and Technology for its internal processes, and the Open Platform enables any associate in the whole company to build new applications.

By rolling out environments for internal use in Product and Technology, the company adopts its own innovative versions first. And by giving associates the ability to create applications, the company gains visibility and feedback for the available IT capabilities.

Many, but not all, Red Hat associates are technical. The IT systems allow for nontechnical users to innovate in their own domains.

More detail on Red Hat's organizational framework for embracing change can be found in *The Open Organization*.

Is there a methodology?

There is no formal methodology to accompany this e-book because the topic of large-scale IT is too difficult to discuss in this manner. Rather, the objective is to draw together a number of key strategic threads related to the way hybrid cloud and cloud-native technologies are impacting organizations.

Thinking of an organization's IT systems as a single application environment with some of these strategy questions in mind brings into focus a number of considerations for the reader:

- Which strategic dimensions will deliver the most gain?
- How do we spread good practice in one area or across the organization?
- How do key types of platforms (operating systems, virtualization, containers) and applications (custom code, integrations, process-driven apps) fit together?
- How do all of these concepts relate to developer productivity and stability for operations?

For a richer analysis of these topics, or an assessment of the areas of greatest impact, please see the next section for programs run by the Red Hat Services teams.

Part five: Conclusions and next steps

IT systems have never been so valuable. However, without a cloud-native, agile strategy that addresses both reliability and productivity, it will become increasingly difficult for businesses to compete in the market. While previous eras of IT have seen great innovations, the past few years have seen an unprecedented reliance on IT to power almost every business function. Back-office and front-office systems must work together like never before.

To succeed in this environment, organizations need a healthy, robust, and productive application environment to provide the foundation for innovation. This application environment needs to span multiple datacenters and clouds while weaving in the latest, most productive cloud-native technologies.

Our objective is to consolidate strategic insights from customers and technology deployments to help others plan and strategize.

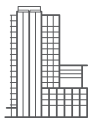
While not all the insights will apply to every scenario, hopefully a number of questions and considerations will be helpful on your IT journey.

Learn more about how Red Hat can support you in your cloud-native and hybrid cloud application journeys:

- See how Red Hat Consulting can help: Get best practice and planning guidance with a Consulting discovery session.
- Check out our Services Speak blog for insights, tips, and more.
- What's your level of DevOps maturity? How ready are you for the cloud-native journey? Take the Ready to innovate assessment to find out.

More information about Red Hat's technology offerings:

- Application delivery: Red Hat OpenShift
- Application development: Red Hat Middleware
- Developer tools: Red Hat Developer Toolset
- Management: Red Hat Smart Management



About Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers integrate new and existing IT applications, develop cloud-native applications, standardize on our industry-leading operating system, and automate, secure, and manage complex environments. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500. As a strategic partner to cloud providers, system integrators, application vendors, customers, and open source communities, Red Hat can help organizations prepare for the digital future.

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

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

redhat.com
#F19145_0919