Modernizing application delivery

Developing a template for modernization success

What is modernization?

Modernization allows organizations to use technologies to enable business goals such as improving efficiency, quality, or differentiating for competitive advantage.

Introduction

While many organizations want to modernize, the task can seem monumental. The systems are large, complex, numerous, and sometimes fragile and barely understood. Application development, security, and delivery—and the way teams work together—are entrenched. The analysis is daunting, and the path forward can seem unclear. These challenges lead to hesitation, delay, and even inaction. While the need for change is unquestioned, there is debate on how to start. In the meantime, however, organizational entropy can ensue and technology debt can accumulate, while the demand for IT resources increases. Market change accelerates, but the pace of digital transformation cannot keep up.

The goals of modernization are unchanged since computing began: faster runtime, stability, security, sustainability, and accelerated delivery of value. For the application development cycle, value means better software with fewer defects and more agile innovation with faster time to market.

Modernization must address people, process, and technology—in tandem. When organizations approach modernization in this way, technology transforms how people work across an organization to deliver business value. By changing the way you work, your application development team can agilely and naturally shift, adapting its workflow iteratively. With continual alignment, it becomes less important how technology changes over time.

Why modernize?

Modernization, by supporting constant change at scale, not only improves processing capacity but also the ability to add value to your business. Value tightly links customer experience to business performance. For example:

- Alliance Bank simplified its branch process for customers to open accounts or add services. By adopting agile development processes, along with Red Hat open source technologies, Alliance Bank achieved 136% year-over-year retail growth while reducing operating costs. Time to market for new features and services is now 2-6 months, down from 18 months. In addition, customer sign-up time shrank by 70%, from 45 to 15 minutes.[1]

[1] Red Hat case study. "Malaysian banking group modernizes branch banking with tablet-based experience," Feb. 2021.

The benefits of modernizing application delivery include:

- Accelerated time to value—bringing features and services to market faster and creating new markets.
- Better client experience—availability and reliability tightly coupled with customer satisfaction.
- Talent retention— reducing the burden on developers and systems administrators and providing a more fulfilling work experience.

- United Parcel Post (UPS) transferred its entire UPS.com infrastructure to a Kubernetes, container-based platform. Modernizing its approach to application development and deployment has been integral to generating and accelerating business value from its technology investments. UPS has improved developer productivity for faster application and feature creation by incrementally delivering services using microservices and containers. The time-to-value has been reduced from over a year to weeks or months.[2]
- Employers created a central application environment with Red Hat® OpenShift® to increase cloud-native and application programming interface (API)-centric development. By standardizing processes and applications in its new environment, Employers improved delivery times and was able to take advantage of new business opportunities. The company's new sales have increased by 40% over three years, and the number of quote requests received and processed has increased by 80%.[3]

Modernization presents an opportunity to improve the quality and types of services offered and the level of innovation. The return on investment is the ability for IT to deliver top-line business and revenue growth by optimizing the client experience and business performance.

Rethink modernization: Dispell assumptions to remove obstacles

When planning a modernization strategy, it is useful to consider common organizational preconceptions, which can inform an organization's idea of modernization and present obstacles to moving ahead.

Preconception #1. Modernization only requires technological change.

There may be practical reasons to change, such as the decision not to renew an enterprise agreement that prompts migration to a new platform. However, if you simply shift your current state to a new provider, then you are just moving your problems to a different infrastructure. Likewise, if your focus is on technology transformation alone, such as adopting cloud and container technologies, without changing the application delivery process—or the teams and individuals responsible for it—how will you accelerate time-to-value?

Sociotechnological systems combine culture, process, and technology as mutually reinforcing elements.

Companies that gain the most value from modernization see their organizations as sociotechnological systems in which culture, process, and technology are interlinked, mutually reinforcing elements. In other words, human systems and technology systems interact, using processes to shape these interactions. Modernization should seek to change the relationship of the interactions between these elements. To allow faster and better applications delivered at scale, each element—culture, processes, and technology—must transform. The result is to create a new operating environment for application development.

Preconception #2. Modernization is a one-time event.

Many organizations approach modernization as a massive undertaking. With this mindset, following a one-time migration comes a period of dormancy in which applications age again. The organization eventually falls back into the same patterns, accumulating new technical debt.

[2]Red Hat case study. "UPS streamlines tracking and delivery with DevOps and Red Hat," Oct. 2018.
[3]Red Hat case study. "Employers centralizes insurance apps on Red Hat OpenShift," March 2021.

By planning for modernization as a continuous improvement process, your organization makes modernization intrinsic to the application development and deployment process. When modernization becomes a constant, it is no longer modernization—it is accelerated digital transformation as continuous value creation. Modernization becomes about iteration, or always beginning. Once you have modernized your application development, you should never have to go through modernization again. It becomes a continuous discipline that supports a culture oriented toward the ability to adapt to change as integral to the normal course of business.

Preconception #3. Modernization is a monolithic undertaking.

The key to organizational transformation as continuous improvement is to achieve quick initial success. At Red Hat, we want clients to avoid large, costly, time-consuming projects that do not produce the intended results. We encourage clients to "fail fast, fail often" for rapid learning and evolution. This approach unlocks value in small, iterative steps that create space for further innovation and modernization by incrementally removing waste from existing processes. We encourage organizations to start small and repeat success.

If something is continuous, the most important action is to start. The idea of modernization is to just begin, picking an area that can have an impact. This approach gives you the evidence to champion success, sell it within the organization, and create demand for more change.

An intentional path to modernization: People+process+technology

There is no one way to modernize application delivery. Each organization undergoes an intentional evolution around people, technology, and process that applies the open source roadmap to its business needs.

1. People: Optimize human systems

Conway's Law states that "organizations will design systems that copy their communication structure."[4] This statement recognizes that human systems and technology systems are interdependent. Therefore, how information flows between groups of people affects a system's design.

If the organization functions in separate segments, the result will be isolated processes and subpar software solutions. It is important to build teams that support the delivery of value streams rather than splitting the value stream among departments separated by function—i.e., a department of developers, of testers, of operators—which splits the value stream into separate pieces. Moreover, when a team is bigger, the interactions become more complex and the communications become less efficient and clear. Smaller teams with tight interlinkage are higher functioning.

From disparate functions to value streams

As you adopt a new system for application delivery, you should consider how to design teams to implement the system. Team topologies, an approach to organizing teams championed by authors Matthew Skelton and Manual Pais, emphasizes the importance of establishing the right patterns of interaction to support fast-flowing software development. Skelton and Pais apply Psychologist John Sweller's concept of cognitive

[4] Gilson, Nathania. "Conway's Law: the little-known principle that influences your work more than you think." Atlassian, Aug. 18, 2021.

load, "the total amount of mental effort being used in working memory" as a key consideration in team design. The goal is to not let the software system grow beyond the cognitive load of its team.[5]

The topology of a team, and getting to the right cognitive load, naturally emerges during the pilot phase of modernization. The team focuses on a value stream and has full autonomy to deliver against a use case.

As the team builds a solution, elements emerge that tax the cognitive load, distracting and slowing the team. For example, the need to provision infrastructure can spin off into a separate team. This establishes a cascade of teams that emerge from the original pilot team, organized around value streams rather than functional segments.

A good example is how security informs the development cycle as part of the value stream rather than being a separate group. DevSecOps integrates security throughout the development cycle. The traditional development life cycle includes a security review late in the development cycle, often just prior green-listing for production. This separate review can be a costly and time-consuming step where delays in delivery generally occur. Building security into your continuous integration/continuous delivery (CI/CD) process means getting feedback about vulnerabilities early and often as the code evolves incrementally.

Autonomy is a key factor in the ability of a team to make effective change. Team topologies focus on maintaining team autonomy while recognizing when the cognitive load around managing a particular aspect becomes too great. In this manner, team topologies evolve.

Empower your technical talent

Included in the implied cost of technical debt is the burden to your developer staff. Your developers may not even understand old programming languages. Coding language becomes an inhibitor towards modernization. Even if your talent pool is stable, its inability to understand languages or other technological aspects can shrink its ability to be agile. By modernizing, you reduce your vulnerability to dated technology and increase the capacity of your existing developers to add value. Training becomes an element of modernization by strengthening organizational knowledge on new tools, practices, and technologies.

2. Technology: Support modernization

Containerized applications, a possible environment for modernization, have all of their runtime dependencies in a single environment, which can then be deployed in different infrastructure environments. The self-service model associated with Kubernetes or containerized applications provides on-demand, at scale reliability coupled with agile development. For example:

- Korean payment card company Lotte Card needed a more agile, responsive environment to better respond to rapid market changes and customer demand by more quickly releasing new features and services. Lotte Card handles more than 3 million transactions daily and must simultaneously process various requests. A cloud-based digital platform, with Kubernetes container technology as the foundation, lets Lotte Card handle concurrent process requests for more than 10x the volume than before and autoscale capabilities to meet the demand of more users. For example, when concurrent users suddenly increased by more than 200%, Lotte Card's applications services continued to operate without disruption.[6]

[5] Skelton, Matthew, and Manual Pais. "Forget monoliths vs. microservices. Cognitive load is what matters." TechBeacon, accessed Feb. 2, 2022.
[6] Red Hat case study. "Lotte Card builds cloud platform with Red Hat OpenShift," July 2020.

- Ascend Money, Southeast Asia's largest financial technology company, replaced its legacy environment with a container solution and developed a standardized approach to application development and delivery. Now Ascend Money builds and releases around 100 apps daily, and this number is growing. The company has transformed the way it delivers software to increase development efficiency. As a result, tasks that previously took one week can now be completed in just two to three days, and the company can now support nearly 200 developers with a technology operations team of only six people.[7]

3. Process: Automate the interaction between people and technology

A container adoption strategy should include how to get the most out of the technology as quickly as possible. The first phase typically starts with a business application, a single metric you want to achieve, or a motivated team that is willing to dive in. The pilot team chooses one or two applications to containerize on the Kubernetes platform quickly.

Choose a candidate for the pilot that is architecturally interesting enough to demonstrate those types of changes, such as moving to a microservice design. For example, the choice could be a flagship application that can provide a real-world working reference for other similar types of applications.

Short and meaningful successes display value and allow the team to internally market success while creating demand within the organization to take on the challenge of modernizing using the new methodology. The pilot team members become champions and internal change agents. The next phase is to make success repeatable for all of the development teams. In the final phase, the organization assumes responsibility for scaling the new approach to application delivery.

How to do an application portfolio assessment

Organizations often have a large inventory of existing software, including applications built on technologies from possibly several decades. You will need to first assess your existing portfolio and determine how to bring it into your new environment. An application portfolio assessment prioritizes applications for modernization and identifies the best migration strategies. The first group of applications identified should offer quick and meaningful success, which other teams can replicate.

An initial step is to identify a subset of applications with shared attributes. This group, based on its common characteristics, associates with a particular migration strategy. The pilot team solves for the identified shared attributes, documents the process, and creates a catalog item for this particular category of applications. The team assigns each category a level of effort based on the ease or difficulty of conversion. This process establishes application archetypes, recurrent patterns that are a reference for other applications in the organization with similar attributes. Categorizing applications becomes repeatable by other individuals and teams, who can follow the pattern for their applications.

Typically the process initially takes a broad and shallow look at applications. Next, there is a slightly narrower and deeper review. Finally, a deep dive looks at applications on a code-level basis. A tiered approach filters out what is not worth investing in or too costly or difficult to migrate and points out where the modernization approach is the obvious choice for delivering the greatest value.

Each category should include a meaningful number of applications. That way, in unlocking one application, the team unlocks all the others in that category. As the team works through a category, it identifies and prioritizes additional groups and then solves for the next most important archetype. The creation of each pattern unlocks more patterns. As each solved pattern overcomes obstacles, there is a diminishing effort.

[7] Red Hat success story. "Ascend Money builds central app program, delivers services faster," accessed Feb. 2, 2022.

At the same time, each pattern derives more value for your organization. In this way, each new batch of applications assessed and migrated benefits from the earlier work.

The organization moves toward rationalizing the portfolio to the fewest common denominators that can be upgraded and standardized. The assessment process helps the organization methodically shift to a container paradigm and restandardize, determining migratable patterns and strategies to address less migratable applications. Through this process, the modernization becomes scalable.

Working through the portfolio assessment and migration moves the organization toward a higher level of maturity for the application delivery process by improving the performance metrics for software delivery. Development activities that were further down in the release cycle move up, and the team identifies opportunities to automate manual processes. This approach should offer more high-quality code at the end of your delivery process.

How to migrate an application

Developers and systems administrators will likely encounter the five Rs of modernization. Each cloud provider and consulting firm defines these terms slightly differently. However, they commonly refer to different application migration strategies.

Repurchase is a decision to buy another vendor's software because of how it is hosted. The decision reflects the need to move from on-premise hosting of a vendor's technology to a vendor-hosted model—or a move from internal hosting on virtual machines to a vendor that uses a container-hosted model.

Rehost moves an application from one server to another without making changes to the application. Rehost is also known as lift-and-shift.

Replatform involves making minor changes to an application so that it can benefit from a cloud-based environment. It involves more work than rehosting and less work than refactoring.

Refactor is the rewriting of an existing application without changing its observable behavior. Refactoring involves varying degrees of work depending on the application. Containerizing an application typically involves some refactoring.

Retire is the process for phasing out an application.

It is important to view the five R's as working together in some combination to support migration as an evolutionary process of continual improvement.

Rationalization (the sixth R) as an organizing principle

Migration should not be a one-for-one mapping of applications from the old environment to the new one. Rationalization, or the consolidation of applications, is an outcome of using the five Rs. For example, India-based Bajaj Allianz Life Insurance (BALIC) consolidated its application development, testing, and deployment onto a single platform in a Kubernetes-based environment. Transforming application components into microservices will enable BALIC to rationalize from 100 monolithic applications to 45 modern applications.[8]

As demonstrated earlier in the portfolio assessment process, rationalization can help plan for and schedule applications for the journey from monoliths to microservices. For example, a number of applications might send alerts via SMS message. Each application has its own implementation of the SMS library to send the message. The goal is not to reimplement the component multiple times in different applications. It is

[8] Red Hat case study. "Indian insurance company adopts microservices to speed digital business," June 2021.

better to rationalize that into a common service. The approach involves rationalizing common business functions into services accessible to multiple end users, eliminating the redundancy of maintaining similar components separately across the organization. A shared-services approach also supports the creation of platform teams organized to support these common components.

Slay a monolithic application in microsteps

The adoption of microservices is a sociotechnical architecture—it involves jointly designing how people and technology work together to optimize the performance of both. Therefore, a microservices approach is a technological outcome of a change in how people work.

Microservices are not necessarily the best solution for all businesses. Business decisions should lead the approach to modernization. As discussed earlier, the process of evaluating the best strategy for your business starts with a domain analysis, or analysis of related software systems in a domain to identify common and variable parts. This approach helps to inform decisions about managing system deconstruction, along with changes to team structures to support your new system. To ensure a thoughtful approach to migration, your organization should review its existing architecture and work toward a modular architecture that can be easily monitored.

Modernizing Java™ applications

If your organization is looking to modernize your monolithic Java applications, using cloud-native and microservice architectures and approaches, read this e-book for guidance on choosing the right path for your application needs.

From monolith to microservices is one migration strategy that illustrates how CI/CD changes in a Kubernetes environment. Simplifying service delivery is a complex design process. When you shift to containers, you disintegrate old ways of delivery (technological systems, human systems, and processes). You integrate your applications in a different way. You integrate functions within IT and between IT and the business in a new way.

The theory of constraint is a methodology developed by Eliyahu Goldratt for identifying the most important limiting factor that is an obstacle to a goal and then systematically eliminating that constraint through improvements.[9] Breaking through bottlenecks allows the optimization of a system's flow. Traditionally, there have always been constraints on how developers build, run, and manage applications in a life cycle. Applying the theory of constraints implies proceeding in a series of small, iterative steps not only for your entire portfolio of applications but for a single application.

The strangler pattern is one migration strategy that illustrates a gradual, incremental migration approach for a single application.[10] In implementing the strangler pattern, the first step is to remove one component and containerize it. This approach leaves the remains of the monolith plus one new microservice. Repeat this step until, by attrition, your monolith becomes small enough to be a container—or disappear completely.

Cloud-native, container-based development

Many organizations adopt a cloud-native approach to create innovative, adaptable, containerized microservices-based applications. A cloud-native architecture horizontally scales small instances to ensure availability, resilience, and reliability. Containers become standard compute units for modern cloud-native applications.

Cloud-native practices should achieve more than optimizing workflows to get change quickly into production. The goal should be to foster creativity and experimentation. Technology is neither a panacea nor the end goal. Rather, technology should catalyze and enable modernization as a way of working. For example, the World Health Organization (WHO) completed a Red Hat Open Innovation Labs virtual residency for ongoing open source innovation.[11] The WHO worked closely with Red Hat to drive new practices alongside the

[9] Theory of Constraints Institute. "Theory of Constraints (TOC) of Dr. Eliyahu Goldratt," accessed Feb. 2, 2022.
[10] Heusser, Matt. "What is the strangler pattern and how does it work?" TechTarget, June 29, 2020.
[11] Red Hat success story. "WHO collaborates with Red Hat Open Innovation Labs to create learning platform," accessed Feb. 2, 2022.

implementation of a platform using open source technology. The WHO Information Management and Technology team was invested in adopting new agile methodology, lean product development, and DevOps practices. The intent of these cultural shifts is to increase adaptivity to changing needs and make the WHO DevOps platform more scalable for current and future demands. During an eight-week virtual residency with Red Hat Open Innovation Labs, the WHO team worked collaboratively with Red Hat experts and gained the knowledge and proficiency to manage its new platform and processes.

A self-service model gives developers and administrators control over the resources to build and manage applications. A cloud-native approach goes further, in that the way those resources are made available and presented is biased toward the needs of application teams. Cloud-native platforms are designed to support container-native development, and thus they provide a level of autonomy. There is greater automation in the life cycle of the applications that transects everything from development to production. The ability to automate how applications run in a live environment requires a level of observability.

By combining self-service resources on demand and an application and value-stream focus, a cloud-native environment helps development teams shift their way of working so that they can improve the cadence of delivery and modernize their applications in a safer, more effective, and automated environment.

Adapt to achieve more

Red Hat Services can help you rehost, replatform, and refactor applications, demonstrating value quickly through a metrics-based approach. At the same time, you will build the organizational capability to onboard teams and applications through training and proven, repeatable processes.

Learn how Red Hat Services can help you:

- Bring the power of cloud-native software delivery to your existing portfolio.
- Build an optimized modernization strategy and plan to limit disruption and focus effort where it is needed most.
- Accelerate and scale modernization through repeatable approaches used throughout your developer community.

Partner with us to build, deploy, and run trusted cloud-native applications anywhere.

NORTH AMERICA
1 888 REDHAT1
www.redhat.com

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com

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

redhat.com
O-F30941