redhat.

THE PATH TO CLOUD-NATIVE APPLICATIONS

8 steps to guide your journey

E-BOOK

TABLE OF CONTENTS

"Digitally advanced enterprises are 8x more likely to grow share but still lag behind digital natives."

Bain Survey: For Traditional Enterprises, the Path to Digital and the Role of Containers

1. SPEED: THE IMPERATIVE FOR DIGITAL BUSINESS

Digital business brings to mind innovative technologies: mobile devices, intelligent sensors, wearable devices, virtual reality, chatbots, blockchain, machine learning, and other technology. For some, it also reflects the rapid rise of new digitally native businesses that have disrupted traditional business models and destabilized established companies and industry sectors. For the majority of organizations, digital business means pivoting to a culture of organizational agility, where the rapid pace of demand can only be satisfied by faster and more flexible development and delivery models. As most organizations do not have the luxury of completely rebuilding their technology foundation or immediately adopting new practices and mindsets, they are embracing gradual yet fundamental shifts in culture, processes, and technology to support greater velocity and agility.

With software increasingly key to how users engage with businesses and how businesses innovate to stay competitive, the speed of application development and delivery is the new digital business imperative.

The cloud-native approach describes a way of **modernizing existing applications and building new applications** based on cloud principles, using services and adopting processes optimized for the agility and automation of cloud computing. This e-book describes detailed steps as a part of a successful journey from where you are today to adopting a cloud-native application approach.

*"'Cloud-native' is an adjective that describes the applications, architectures, platforms/infrastructure, and processes, that together make it **economical** to work in a way that allows us to **improve our ability to quickly respond to change** and **reduce unpredictability.**"*

CHRISTIAN POSTA
CHIEF ARCHITECT AT RED HAT
AND AUTHOR OF *MICROSERVICES FOR JAVA DEVELOPERS*

SOURCE: INFOQ, "DEFINING CLOUD NATIVE: A PANEL DISCUSSION," 2017.

2. WHAT IS A CLOUD-NATIVE APPLICATION?

A cloud-native application is an application built to take advantage of cloud computing models to increase speed, flexibility, and quality, while reducing deployment risks. Despite its name, a cloud-native approach is not focused on ***where*** applications are deployed, but instead on ***how*** applications are built, deployed, and managed.

Cloud-native approaches are similar to microservices architectures. However, although microservices can be one of the outcomes of building cloud-native applications, there are many steps to reach the level of maturity for managing microservices in production. Microservices are not required to take advantage of all benefits provided by cloud-native apps. Many organizations achieve the benefits of cloud-native approaches by focusing on building better modular monoliths using the same principles.

Evolving toward cloud-native application development and delivery is multidimensional, affecting culture, processes, architecture, and technology. As such, this is a journey rather than a destination, representing a cycle of change that can be challenging to embrace.

3. TRADITIONAL VERSUS CLOUD-NATIVE APPLICATIONS

The differences between cloud-native application development and traditional application development highlight facets of necessary change.

TABLE 1. TRADITIONAL VERSUS CLOUD-NATIVE APPLICATION DEVELOPMENT

	TRADITIONAL	CLOUD-NATIVE
FOCUS	Longevity and stability	Speed to market
DEVELOPMENT METHODOLOGY	Waterfall, semi-agile development	Agile development, DevOps
TEAMS	Isolated development, operations, QA, and security teams	Collaborative DevOps teams
DELIVERY CYCLES	Long	Short and continuous
APPLICATION ARCHITECTURE	Tightly coupled Monolithic	Loosely coupled Service-based Application programming interface (API)-based communication
INFRASTRUCTURE	Server-centric Designed for on-premise Infrastructure-dependent Scales vertically Preprovisioned for peak capacity	Container-centric Designed for on-premise and cloud Portable across infrastructure Scales horizontally On-demand capacity

3.1 TRADITIONAL APPLICATION DEVELOPMENT AND DELIVERY

Many applications that are fundamental to business operations were not designed with digital experiences in mind. Characterized by long lifespans, they were built as tightly coupled monoliths, built over a period of time to well-defined specifications that were often determined long before delivery.

These development approaches were largely waterfall and sequential, spanning long periods of time, and only more recently combined with semi-agile practices. The stages of application development, testing, security compliance, deployment, and management were isolated into functional areas with distinct teams, roles, and responsibilities, with linear communication flows between parties.

These applications were built as large, multifunctional, tightly coupled applications, where a user interface, various application services, code to access data, and other components were combined in a single application, regardless of the technology environment. For example, an e-commerce application built as a tightly coupled monolith would generally include all the functionality for the web user interface, product catalogs, shopping cart, product recommendations, product ratings and review, payment system, and other components needed to make purchases on the e-commerce website—all in one application.

For the majority of traditional applications, infrastructure was preprovisioned for the peak capacity required for the applications, and scaling was achieved by increasing the hardware capacity of the server through vertical scaling.

By 2020, more than 50% of Mode 1* applications migrated from private datacenters to the public cloud will be rewritten using cloud-native architectural precepts, up from less than 10% in 2017.

Gartner: Why You Must Begin Delivering Cloud-Native Offerings Today, Not Tomorrow, January 2018

3.2 CLOUD-NATIVE APPLICATION DEVELOPMENT AND DELIVERY

With a focus on speed to market, cloud-native application development requires more agile, service- and API-based development and continuous delivery approaches. These capabilities are supported by DevOps collaboration across development and delivery teams, more modular architecture, and flexible infrastructure that can scale horizontally on demand, support multiple environments, and offer application portability.

With the flexibility and agility offered by modern cloud technologies, organizations want to move traditional applications to cloud environments to take advantage of greater agility and on-demand compute capacity.

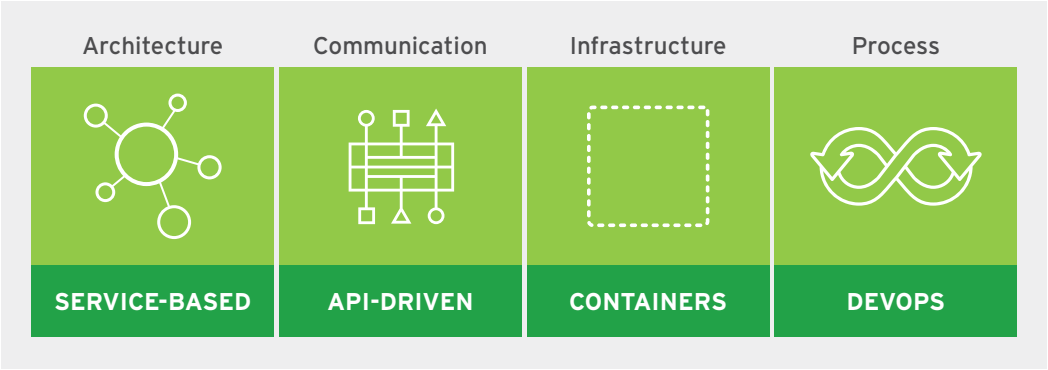
However, many of the operational capabilities built into traditional platforms are either obsolete and not required in a cloud environment, or they are simply provided and operationalized by the cloud environment itself. Cloud environments simplify management of the host's life cycle, as well as helping organizations take advantage of immutable infrastructure principles and tune hosts to the needs of a single application instance.

The path to cloud-native applications can vary by organization. Just creating microservices does not lead to the service quality and delivery frequency required by digital business. Likewise, just adopting tools that support agile development or IT automation will not lead to the increased velocity of cloud-native approaches. Rather, it is a combination of practices, technologies, processes, and mindsets that will define success.

* ***Gartner defines bimodal*** *as the practice of managing two separate but coherent styles of work: one focused on predictability; the other on exploration.* ***Mode 1 is optimized for areas that are more predictable and understood. It focuses on exploiting what is known, while renovating the legacy environment into a state that is fit for a digital world.*** *Mode 2 is exploratory, experimenting to solve new problems and optimizing for areas of uncertainty.*

4. FOUR TENETS OF CLOUD-NATIVE APPLICATION DEVELOPMENT AND DEPLOYMENT

Cloud-native application development is an approach to building and running applications that takes full advantage of the cloud computing model based upon four key tenets: Service-based architecture, API-based communication, container-based infrastructure, and DevOps processes.



SERVICE-BASED ARCHITECTURE

Service-based architecture, such as microservices, advocates building modular, loosely coupled services. Other modular architecture approaches—e.g., miniservices—that respect loose-coupling and service-based design help organizations increase application creation speed without increasing complexity.

API-BASED COMMUNICATION

Services are exposed via lightweight, technology-agnostic APIs that reduce the complexity and overhead of deployment, scalability, and maintenance. Businesses can create new capabilities and opportunities internally and externally via the exposed APIs.

API-based design only allows communication via service interface calls over the network, avoiding the risks of direct linking, shared memory models, or direct reads of another team's datastore. This design extends the reach of applications and services to different devices and forms.

CONTAINER-BASED INFRASTRUCTURE

Cloud-native applications rely on containers for a common operational model across technology environments and true application portability across different environments and infrastructure, including public, private, and hybrid. Container technology uses operating system virtualization capabilities to divide available compute resources among multiple applications while ensuring applications are secure and isolated from each other.

Cloud-native applications scale horizontally, adding more capacity by simply adding more application instances, often through automation within the container infrastructure.

The low overhead and high density of containers, allowing many of them to be hosted inside the same virtual machine or physical server, makes them ideal for delivering cloud-native applications.

DEVOPS PROCESSES

Application development for cloud-native approaches follows agile methods with continuous delivery and DevOps principles that focus on building and delivering applications collaboratively by development, quality assurance, security, IT operations, and other teams involved in delivery.

More than half (51%) of large organizations have already adopted DevOps. However, most currently use DevOps for only 10-40% of apps (20% typical).

IDC PaaS View for the Developer Survey, November 2017

5. THE PATH TO CLOUD-NATIVE APPLICATIONS: 8 STEPS

STEP 1: EVOLVE DEVOPS CULTURE AND PRACTICES

The path to cloud-native applications requires development and IT operations teams to evolve in many different ways to build and deploy apps faster and more efficiently. Regardless of industry or size, every business needs to consider the wide range of activities, technologies, teams, and processes that together form a DevOps culture. To take advantage of new technology, faster approaches, and tighter collaboration, organizations must truly embrace the principles and cultural values of DevOps and organize themselves around those values.

With the complexity of managing multiple distributed environments, highly customized legacy applications, and new application workloads in an era of rapid digital innovation, DevOps can be challenging for some organizations. There is still untapped potential in broadening DevOps practices across the application portfolio.

The adoption of a DevOps culture relies not just on tools and technologies, but also on the willingness and trust of people to embrace a more integrated and collaborative approach to developing and delivering applications. The culture of open source software projects can be a guide to building a DevOps culture.

In Red Hat Open Innovation Labs, organizations are guided in the DevOps process to encourage experimentation, fast failure, transparent decision-making, and using recognition and rewards to boost trust and cooperation. In this environment, designed to catalyze innovation, teams use innovative open source technologies to rapidly build prototypes, experience DevOps, and adopt agile workflows.

Read more about how Red Hat Open Innovation Labs can help your DevOps journey
DOWNLOAD THE E-BOOK

"If you can't build a well-structured monolith, what makes you think you can build a well-structured set of microservices?"

SIMON BROWN
CODINGTHEARCHITECTURE.COM/PRESENTATIONS/SA2015-MODULAR-MONOLITHS

STEP 2: SPEED EXISTING APPLICATIONS USING FAST MONOLITHS

When embarking on a cloud-native application journey, organizations should not only focus on new development. Many legacy applications are critical to business operations and revenue generation and cannot simply be replaced. Rather, they need to be integrated with new cloud-native applications. But how do you speed up an existing monolith? The answer is to take a fast monolith approach by moving your existing monolithic architecture to a more modular, service-based architecture and API-based communication.

Before beginning the onerous task of refactoring monolithic applications into microservices, organizations should first create a solid foundation for their monolithic architecture. Although monolith applications are associated with lack of agility, their poor reputation is mostly due to the way these monolithic applications are built. A fast monolith, however, can achieve many of the agile benefits associated with microservices—without the added complexity and costs.

Evaluating a fast monolith approach ensures that applications are built following solid design principles and properly defined domain boundaries. This approach supports a more gradual and less risky transition to a microservices architecture, if needed. Evolving a fast monolith in this way sets the foundation for a successful microservices architecture.

If applications were not designed using a fast monolith approach, they can still be made faster by moving the existing monolith to a container-based platform. This shift speeds deployment and delivers higher return on investment (ROI). Subsequent integrations or features for the monolith can be built using cloud-native techniques and approaches.

You can also start breaking down your monolith into smaller components at your own pace, using a phased approach.

STEP 3: USE APPLICATION SERVICES TO SPEED DEVELOPMENT

Reusability has always been key to speeding software development, and cloud-native applications are no exception. However, reusable components for cloud-native applications must be optimized and integrated into the underlying cloud-native infrastructure to fully provide the benefits.

Why re-create a caching service, rules or workflow engine, integration connectors, mobile and API management capabilities, data virtualization service, messaging broker, or serverless framework when you can use existing ones that have been optimized and integrated to the underlying container-based infrastructure? These application services, whether they are Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), or iPaaS offerings, are effectively ready-to-use developer tools.

Cloud-native applications may need one or more of these type of services to help developers accelerate development and get new applications to market faster. Whereas DevOps and containers accelerate the delivery and deployment of a cloud-native application, application services accelerate its development.

For example, cloud-native application developers can take advantage of application services specifically built to not only perform well in a container-based infrastructure, but also to take advantage of platform capabilities, such as CI/CD pipelines, rolling and blue/green deployment, automatic scalability, fault tolerance, and more.

STEP 4: CHOOSE THE RIGHT TOOL FOR THE RIGHT TASK

An increase in the fields of software study, such as Internet of Things (IoT), machine learning, artificial intelligence (AI), data mining, image recognition, self-driving cars, and more, has resulted in a growing variety of frameworks, languages, and approaches for software development.

Building cloud-native applications is becoming more diverse as the choice of language or framework is increasingly tailored to the specific business application need. The resulting increase in complexity merits use of a container-based application platform that supports the right mix of frameworks, languages, and architectures to support cloud-native development.

Cloud-native development also requires choosing the right tool for the right task. Whether cloud-native applications are being implemented using a 12-factor approach, domain-based design, test-based design and development, MonolithFirst, fast monolith, miniservices, or microservices, the cloud-native platform must offer the right mix of frameworks, languages, and architectures to support the chosen development requirements. In addition, the underlying container-based platform should support a set of curated runtimes and frameworks that is continuously updated in line with technological changes.

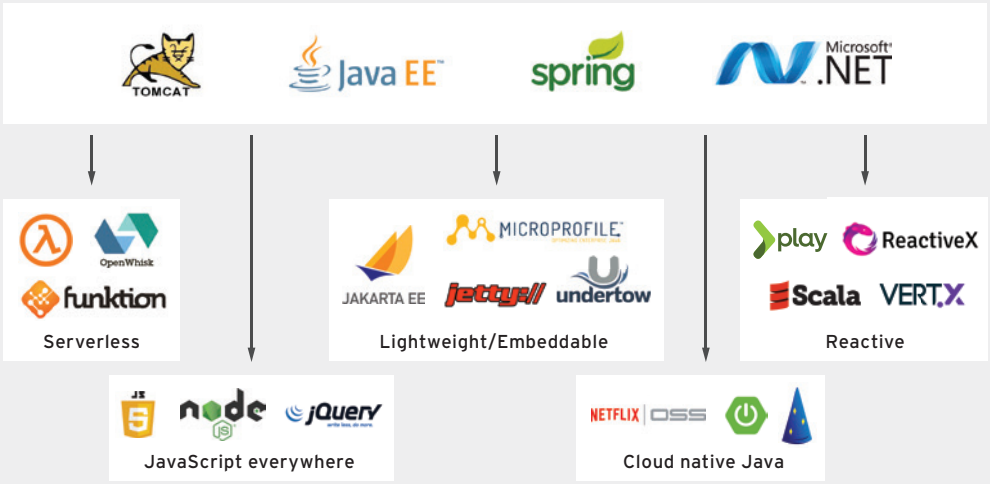


Figure 1. Cloud-native application development becomes even more diverse

STEP 5: PROVIDE SELF-SERVICE, ON-DEMAND INFRASTRUCTURE

Agile methods have helped developers create and update software quickly but lack an efficient mechanism for timely infrastructure access when and where it is required. When releasing applications to production, overall speed to market is affected. Filing a ticket and waiting weeks for IT operations to release resources is no longer a sustainable model in an era when infrastructure is inexpensive and engineering talent is costly.

Self-service and on-demand infrastructure provisioning provides a compelling alternative to unauthorized shadow IT by allowing developers to access the infrastructure they need, when they need it. But this model can only be effective if IT operations teams have control and visibility across what is often a dynamic and complex environment.

Containers and container orchestration technology abstract and simplify access to underlying infrastructure and provide robust application life-cycle management across various infrastructure environments, such as datacenters, private clouds, and public clouds. A container platform offers additional self-service, automation, and application life-cycle management capabilities. This model lets developers and operations teams spin up consistent environments quickly, helping developers focus on building applications without the obstacles and delays associated with provisioning infrastructure.

Standardization is also an important part of a self-service model. It helps organizations automate and deliver consistently to meet business objectives. Process standardization involves mapping the exact sequence of events and activities needed to achieve a task—for example, releasing an application to a new environment.

Containers also support application portability, including the creation of a cloud-native application that can be deployed and run on any cloud provider. Portability offers the freedom to select any cloud provider at any point in time, easily migrate from one cloud provider to another, optimize related costs, and develop a multicloud application without coding to a specific cloud provider API.

Learn more about different practices and techniques that support the cloud-native journey.
DISCOVER THE OPEN PRACTICE LIBRARY

STEP 6: AUTOMATE IT TO ACCELERATE APPLICATION DELIVERY

IT or infrastructure automation is essential to accelerating the delivery of cloud-native applications by eliminating manual IT tasks. Automation can integrate with and apply to any task or component, from network and infrastructure provisioning to application deployment and configuration management.

IT management and automation tools create repeatable processes, rules, and frameworks that can replace or reduce labor-intensive human interaction that delay time to market. They can extend further into specific technologies, like containers, or methods, like DevOps, into broader areas, such as cloud computing, security, testing, monitoring, and alerting. As a result, automation is key to IT optimization and digital transformation, speeding overall time to value.

Guides for IT automation

1. Adopt an enterprise-wide, programmatic automation approach to IT operations. Embrace collaborative dialog across the organization to design service requirements.
2. Consider automation sandboxes as the foundation for learning the automation language and processes.
3. Think hard about automation. Make sure every unnecessary manual step is eliminated, even if it is tempting to retain manual controls for peace of mind.
4. Consider tackling automation incrementally in small, achievable steps using systematic methods. Each step builds on the previous one to create a widespread automation practice.
5. Start by automating one task or service—whether compute, network, storage, or provisioning. Share that automation with others and build upon it systematically.
6. Implement self-service catalogs that empower users and speed delivery.
7. Implement metering, monitoring, and chargeback policies and processes.

Over time, integrated, full-scale automation will not only become a reality, but will yield higher efficiency, faster DevOps, and rapid innovation.

Learn more about the important role of IT automation in "The automated enterprise"
DOWNLOAD THE E-BOOK

Continuous delivery (CD) is a software engineering approach where teams keep producing valuable software in short cycles while ensuring that the software can be reliably released at any time. Through reliable, low-risk releases, CD makes it possible to continuously adapt software to incorporate user feedback, shifts in the market, and changes to business strategy.

Definition from Gartner

STEP 7: IMPLEMENT CONTINUOUS DELIVERY AND ADVANCED DEPLOYMENT TECHNIQUES

Long release cycles mean longer delays between the discovery and resolution of software bugs, as well as an inherent barrier to timely responses to customer and market demand changes. For high-traffic applications—such as mobile, web, or IoT applications—an unresolved bug can affect many users, resulting in poor customer experiences, security or safety issues, and reduced productivity or revenue. Even for other internal business applications, outages or delays in addressing software bugs can have high business costs.

Agile development methods evolved to create a model of release early, release often. DevOps and continuous delivery approaches extend these methods by closely uniting developers, operations, quality assurance, and security teams to improve software delivery processes. As a result, code changes can be pushed to production quickly and reliably to provide fast feedback to developers. This iterative, fast feedback loop is enabled through CI/CD, extending infrastructure automation to an end-to-end, automated delivery system that covers all aspects of application delivery, including automated testing, vulnerability scanning, security compliance, and regulation checks. The goal of automated delivery pipelines is to provide updates without affecting operational capacity, reducing delivery risks.

The first step in achieving continuous delivery (CD) is to enable continuous integration (CI). CI systems are build systems that watch various source control repositories for changes, run any applicable tests, and automatically build the latest version of the application from each source control change, such as Jenkins.

See how some modern automation technologies like Red Hat Ansible® Automation support CI/CD
DOWNLOAD THE WHITEPAPER

"Advanced deployment techniques bring structure and clarity to innovation. Mature deployment methodologies create an environment that allows true experimentation, feedback, and analysis. Better experimentation leads to better innovation."

BURR SUTTER
DIRECTOR OF DEVELOPER EXPERIENCE, RED HAT
REDHAT.COM/EN/ENGAGE/TEACHING-AN-ELEPHANT-TO-DANCE

Advanced deployment patterns aim to reduce the risk of software releases and build an environment for experimentation with controlled outcomes without unintended negative consequences for customers. This goal is essential for increasing innovation across an organization.

Advanced deployment techniques change the nature of delivery from an off-hour weekend activity, with service windows and down times, to a routine workday activity with zero downtime in production, while the application is still available to the customers.

By removing the inconvenience of new deployment for the customers, these techniques let organizations deliver updates and releases at the frequency that business demands. The following are some of the common deployment techniques that can be used to achieve zero-downtime deployment, depending on the application use cases:

Rolling deployment is a pattern where, instead of updating all instances of an application at once, each instance is updated individually by excluding it from the load balancer so that it does not receive traffic. It is updated and then included again in the load balancer. This process continues until all instances are updated.

Blue/green deployment describes the practice of running two identical environments, one active and the other idle. Changes are rolled out to the idle environment, then, once the change is verified in production, the live traffic is switched to the updated environment. Rolling back to the previous version is as simple as switching the traffic back, provided that the data transition is also taken into consideration.

Canary deployment is similar to blue/green deployment in that it uses two identical environments. However, it differs in the way roll out is controlled. After deploying a new release, a small subset of customers are sent to the new release to test it in production. If the new release verification succeeds, traffic is incrementally shifted to the new version while the outcomes are monitored and verified until all users are sent to the new release.

Teaching an elephant to dance
DOWNLOAD THE E-BOOK

STEP 8: EVOLVE A MORE MODULAR ARCHITECTURE

In a microservices-based architecture approach to writing software, applications are broken down into their smallest components, independent from each other. Instead of a traditional, monolithic approach where everything is built into a single piece, microservices are separated components that work together to accomplish the same tasks. This approach to software development values granularity, being lightweight, and sharing similar process across multiple apps. Although a microservices architecture does not impose a specific underlying infrastructure, a container-based platform an optimal foundation.

Evolving a microservices-based architecture might provide an extra benefit for very large teams or production deployments multiple times a day. From an architectural standpoint, microservices requires breaking out each service into its own deployment unit. Each microservice is then managed and deployed independently, potentially with different teams responsible for their life cycles.

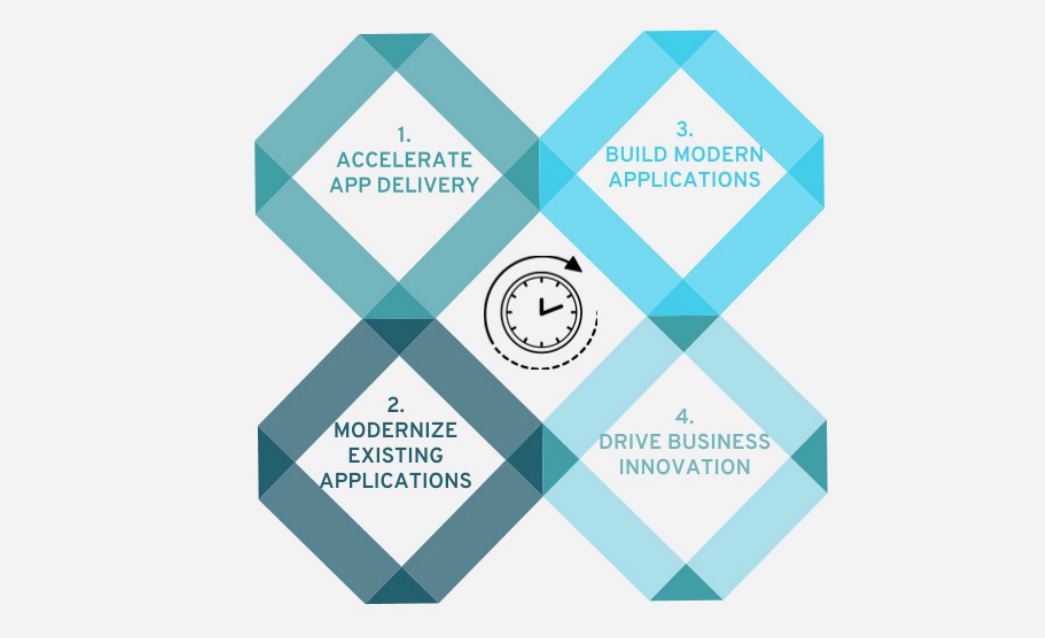
However, implementing a microservice architecture requires investment and skills and may prove too disruptive to an organization. Analysts and subject-matter experts recommend taking a MonolithFirst approach to microservices, which means building an application as a monolith first, even when your intention is to create a microservices architecture. The purpose of this endeavor is to first understand the domain of your application well and then better recognize bounded contexts within it that would serve as candidates to convert into microservices. By doing so, this approach avoids technical debt, such as repair costs when building a set of microservices before understanding the domain and bounded contexts of the application.

Another alternative to microservices is miniservices. A miniservice is a collection of services that are split by domain and usually run on an application server. Miniservices improve agility and scale without the complexity of microservices-based design and infrastructure. Miniservices still require an investment in agile, DevOps, and CI/CD approaches, making a modern application server or a multi-framework, -architecture, and -language offering in combination with a container-based infrastructure ideal.

A platform that supports different frameworks, languages, and approaches to cloud-native application development—e.g., microservices, miniservices, or MonolithFirst—is key to success with cloud-native applications.

6. BUSINESS CASES FOR CLOUD-NATIVE APPLICATIONS

Businesses will have different priorities in their digital transformation efforts. Some are modernizing their existing application architecture and infrastructure as they evolve toward modern, cloud-native principles, while others are innovating with new business models and applications. Whatever the intent and business use case, they all share the objectives of greater speed, flexibility, and digital readiness. The common use cases for cloud-native applications can be categorized broadly under the following four business challenges:



*"On **day 1** of our acquisition of a new bank, **10 changes** were put into production, with **zero defects.**"*

JOHN RZESZOTARSKI
DIRECTOR OF DEVOPS, KEYBANK

Reduction in deployment time:
12 to 1 weeks

BUSINESS CHALLENGE #1: ACCELERATE APPLICATION DELIVERY

Objective:
Increase the speed of **application delivery** of existing and new applications to customers.

Approach:
Containers provide a common platform that unites development, operations, security, QA, and other teams to embrace DevOps independent of infrastructure and application technology. With a DevOps approach, teams use automation and CI/CD practices to release software quickly and confidently. By addressing deployment issues through container-based automation, the app delivery cycle can be accelerated and adapted to the pace of business, rather than adapting the pace of business to what IT can deliver.

KeyBank

Customer spotlight:
KeyBank, one of the top 15 banks in the United States, launched a digital channel modernization initiative to update its web experience and create a new mobile web app. With Red Hat OpenShift® supporting its migration from monolithic applications to microservices, KeyBank built an automated continuous delivery pipeline and moved from quarterly to weekly deployments.

By 2023, 90% of current applications will still be in use, but most will have received insufficient modernization investment.

Gartner: Application Modernization should be business-centric, continuous, and multiplatform, January 2018.

"With expert support for Red Hat JBoss Enterprise Application Platform, we don't have to worry about day-to-day operations."

FORMER GENERAL MANAGER
CORPORATE TECHNOLOGY,
AUSTRALIAN SECURITIES EXCHANGE

60x faster application restart speeds

Decreased platform support costs and time, freeing resources for innovative service development

BUSINESS CHALLENGE #2: MODERNIZE EXISTING APPLICATIONS

Objective:
Increase the speed of change by **modernizing existing applications** to adapt to the markets and customers.

Approach:
Many valuable business applications are legacy applications that were not designed for the digital era. However, a rip-and-replace approach is not always feasible or economically viable. In addition, not all legacy applications lend themselves to modernization.

When the migration of traditional applications to the cloud is considered viable, this approach is supported by containers, where dependencies on the underlying infrastructure can be removed. As a result, applications are portable from on-premise infrastructure to the cloud where they may also, if needed, be refactored and re-architected to become cloud-native. A container platform approach can also take advantage of the platform's automation capabilities and DevOps practices to simplify migration of existing applications.

Customer spotlight:
As the first major financial market to open each day, the Australian Securities Exchange (ASX) plays a significant role in the global financial services sector. The organization must operate with high stability, security, and performance, but its legacy application server platform was becoming increasingly inconsistent, unstable, and expensive. ASX established an initiative to modernize its digital platform with new technologies and decided to deploy Red Hat JBoss® Enterprise Application Platform to create a robust foundation for its application server. The initial deployment involved one of the organization's critical business-to-business (B2B) web applications, ASX Online, that provides prices, company announcements, and critical reporting to the market while meeting regulatory requirements.

BUSINESS CHALLENGE #3: DEVELOP NEW CLOUD-NATIVE APPLICATIONS

Objective:
Increase the speed of **developing new applications** to address new business opportunities.

Approach:
Changes in business and customer demand present opportunities for organizations that can quickly turn ideas into services and products, evaluate their outcome in the new landscape, and then adapt. The cloud-native approach to building new applications is a step toward accelerating the path from idea to innovative applications with support from service-based architecture, API integrations, containerized services and orchestration, and DevOps practices, automation, and tools.

"Red Hat OpenShift Container Platform has truly stolen my heart. It is innovative and lets us deploy quickly and easily control our containers."

MICHAEL AALBERS
SENIOR TECHNICAL APPLICATION COORDINATOR, AMSTERDAM AIRPORT SCHIPHOL

50% faster development of new APIs

Schiphol Amsterdam Airport

Customer spotlight:
Schiphol International Airport is the third busiest in Europe with 64 million passengers per year. Schiphol's goal is to become the world's best digital airport by 2018. To achieve this goal, it needed to accelerate application development through a cloud-agnostic platform. One key part of Schiphol's digital strategy is services delivered via APIs, including its Flight API, which provides passengers with information such as gate, terminal, and check-in time. With Red Hat OpenShift Container Platform, Schiphol is creating a self-service, multicloud platform for its internal IT team and business partners, cutting development time for new services.

BUSINESS CHALLENGE #4: DRIVE BUSINESS INNOVATION

Objective:
Increase the **speed of innovation** across the organization to the pace that business demands.

Approach:
In a fast-moving world, standing still means falling behind. IT teams are racing to quickly introduce new features and services that delight customers and help employees work smarter. Success hinges on constant innovation and relies on more than just new tools and technologies. It requires a new culture, tools, and processes that use revolutionary perspective to support innovation and experimentation across the organization.

"The most exciting part of the engagement is that we're going to re-engineer the way IT works. We're going to totally change the way that we work as a business, [and] we're going to start the process of changing the way the whole bank works."

WAYNE MARCHANT
CHIEF INFORMATION OFFICER, HERITAGE BANK

Heritage Bank

Customer spotlight:
Heritage Bank is 142 years old, one of Australia's oldest financial institutions. Facing increasing competition in the market and new market mandates, Heritage Bank needed to find new ways to deliver software faster. Through team immersion in Red Hat Open Innovation Labs, Heritage Bank created an innovative banking solution alongside a high-performing team, a team that is now able to continue to develop better software, more quickly, well into the future.

Video of Heritage Bank

HOW RED HAT CAN HELP

Depending on your stage within a digital and cloud-native journey and your priorities, Red Hat has the technologies and services to support you.

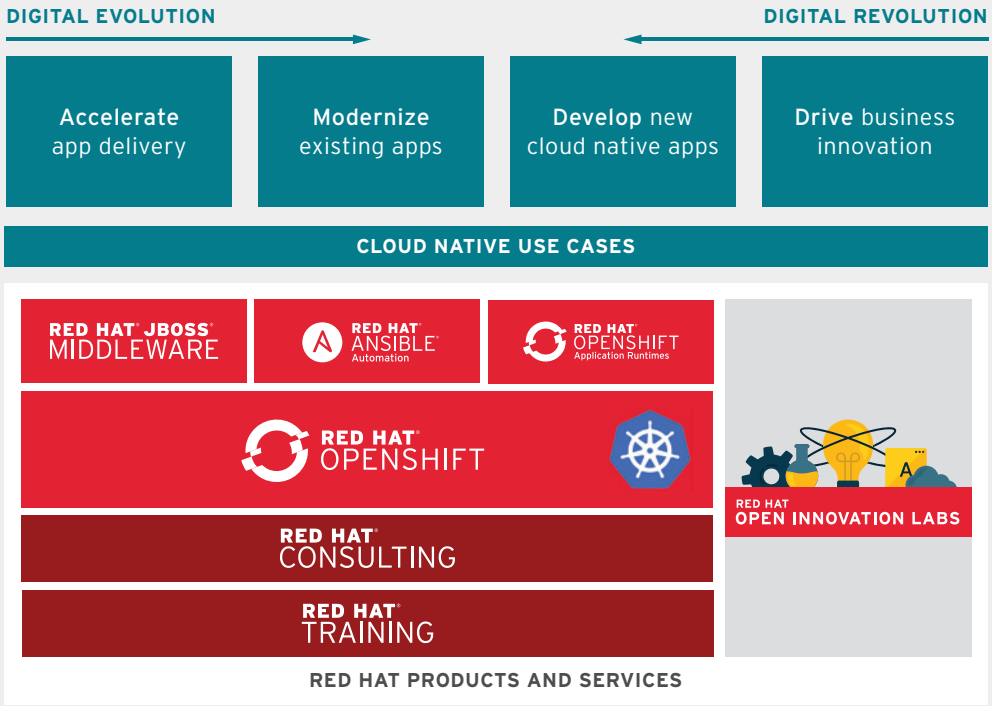


Figure 2. The cloud-native journey with Red Hat

Some organizations may be focused on only one cloud-native use case, while others may be prioritizing a few use cases simultaneously. Whether you take an evolutionary or a revolutionary approach, your path is highly individual and not necessarily linear. Whatever your path, getting applications to market faster requires the right technology, DevOps practices, and culture.

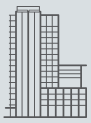
Red Hat helps support this journey with Red Hat OpenShift, a cloud-native container development platform. Red Hat OpenShift Application Runtimes offers open source runtimes and frameworks for building cloud-native applications, accelerating development time via containerized runtimes on OpenShift. A range of Red Hat JBoss Middleware technologies can be deployed on OpenShift, including Ansible automation and management technologies.

To help navigate the complexity of digital transformation, Red Hat Consulting offers strategic advice as well as in-depth technical expertise. From Red Hat Open Innovation Labs to Discovery Sessions and project implementation plans, our consultants can help you on every step of your cloud-native journey.

ARE YOU ON A CLOUD-NATIVE JOURNEY?

Learn more about how Red Hat can support your cloud-native application journey:

- 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.



ABOUT RED HAT

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

NORTH AMERICA
1 888 REDHAT1

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

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

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

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

redhat.com
#F12255_0518