New Relic.

Cloud Native Is the New Normal

Is Your Environment Optimized for Success?

 | US +888-643-8776 | www.newrelic.com | www.twitter.com/newrelic | blog.newrelic.com

Table of Contents

Introduction

The world has moved on, again.

In a relatively short span of time, we've evolved from the cloud age, in which the resources needed to run applications are available as a service in the cloud, to the cloud native age, in which the applications running in the cloud are purpose built and optimized to take full advantage of cloud benefits such as elasticity and resiliency.

It seems that everyone in the industry is talking about cloud native—including the role of distributed systems, microservices, containers, serverless computing, and other emerging technologies and architectures. While many organizations are somewhere in the process of defining and optimizing their cloud approach, there's still a fair amount of confusion among IT practitioners and decision makers about what cloud native really means and what the ultimate goals and expectations should be for an organization leveraging modern technologies and practices.

The biggest question of all may be "How can my organization be successful with a cloud native approach and what impact will this successful technology approach have on my business objectives?"

This ebook aims to answer this fundamental question by relating the technical and cultural initiatives essential to the cloud native journey back to how they drive positive outcomes for both IT and the business. We'll also sprinkle in some best practices for extracting maximum value out of your cloud environment.

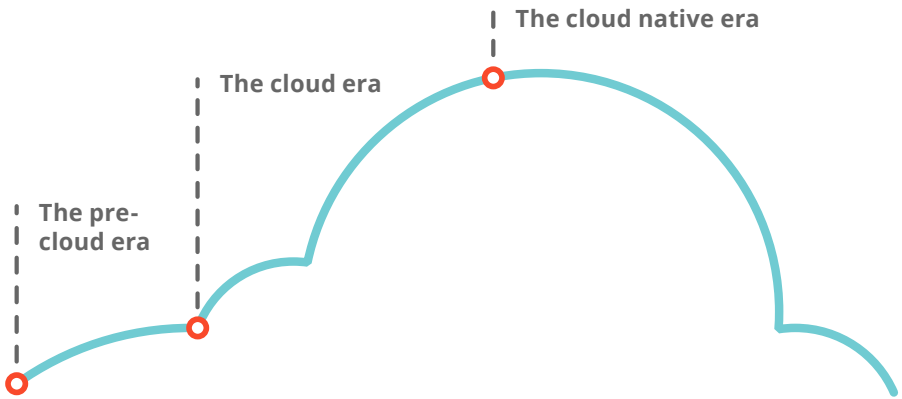
> *"From an economic point of view, cloud native technologies enable the true value of cloud by allowing applications to scale and evolve in much shorter timelines than previously. This scalability creates new opportunities for the business in terms of revenue growth, efficiency improvements, or a better customer experience."* [1]

1: "The 451 Take on cloud native: Truly transformative for enterprise IT," 451 Research LLC, 2019

CHAPTER 1

How We Got Here

Chapter 1: How We Got Here



Before we get into what cloud native means and how to optimize the value we get from modern software systems, let's review how we got here.

The pre-cloud era

- **Software development:** Using a waterfall approach, software took years to create, with infrequent releases of new features. Development, quality assurance, and operations were distinct groups, often operating in silos.
- **Infrastructure:** Physical hardware was located on-premise, and as server virtualization took hold, virtual servers were deployed to consolidate operations on fewer, larger physical servers.
- **Operations:** Operational tasks were highly manual, limiting the scale at which systems could operate. Over time, provisioning and maintaining the physical and virtual infrastructure were made less onerous by configuration management tools.

The cloud era

- **Software development:** Agile development and DevOps approaches took hold as alternatives to the waterfall approach for faster development of higher quality software.
- **Infrastructure:** The public cloud made resources needed to run applications available as a service. Companies embraced the cloud as a way to focus on their core business and offload portions of their IT infrastructure, paying only for the resources consumed.
- **Operations:** Automation became more prevalent for cloud operations and management, including provisioning, configuration, scaling, and self-healing.

The cloud native era

- **Software development:** Application monoliths are being replaced with new software architectures such as microservices and serverless functions, and taking advantage of the key characteristics of cloud computing.
- **Infrastructure:** Platforms are available as a service in the cloud. Organizations often adopt a multi- or hybrid-cloud approach, deploying applications and services wherever it makes the most sense to run them.
- **Operations:** Software running on distributed systems is enabled by new deployment methods such as containers and container orchestration platforms. Increased abstraction in applications and infrastructure means developers approach system health and performance in new ways, taking advantage of improvements while grappling with increased complexity.

This progression toward the cloud native era highlights the fact that a simple migration or "lift and shift" of applications to the cloud is not enough to exploit the true benefits of the cloud. A 2018 study by DORA (DevOps Research and Assessment team) showed that how organizations adopt the cloud matters, and teams who met the following requirements were more likely to be elite performers[2]:

- Adopted essential cloud characteristics (on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service)
- Leveraged infrastructure as code and modern deployment and pipeline practices
- Took advantage of platform-as-a-service (PaaS)
- Adopted cloud native design practices

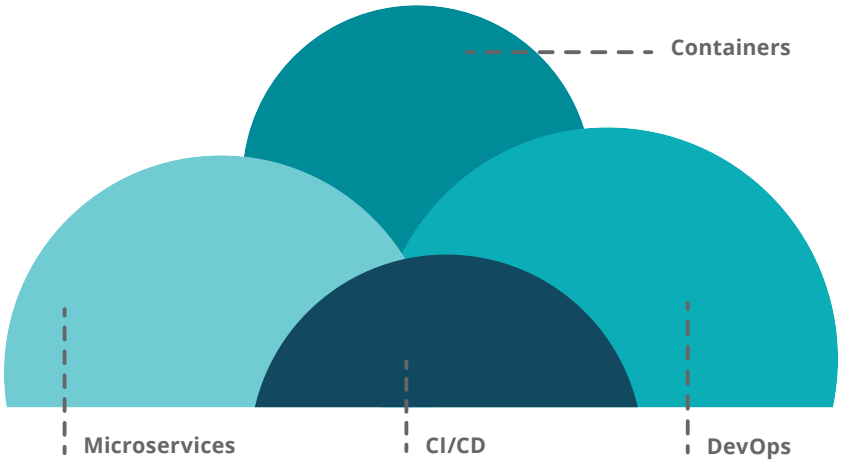
Which brings us to the discussion of what cloud native means and why it's critical to helping organizations achieve their business objectives.

2: "Accelerate: State of DevOps, Strategies for a New Economy," DORA, 2018

CHAPTER 2

Welcome to the Cloud Native Era

Chapter 2: Welcome to the Cloud Native Era



Even though many organizations are just getting started with their cloud native initiatives, it's already become one of the biggest trends in the industry. The Cloud Native Computing Foundation reports that the use of cloud native technologies in production grew more than 200% between December 2017 and August 2018[3]. A survey from Capsule8, Signal Sciences, and Duo shows that 62% of companies are relying on cloud native technologies for more than half of their new applications. That figure is expected to grow to more than 80% by 2021[4].

Before we dive into why cloud native technologies and practices are exploding so quickly, let's first start with some definitions.

Describing cloud native

While there is no single, standardized statement that explains what cloud native means, the essence of the term and how it is used is this: Cloud native is about optimizing applications and their environments to take maximum advantage of the core characteristics of cloud computing with the goal of achieving transformational digital and business outcomes.

Applications that are built to run in cloud environments are:

- Designed for and deployed to distributed computing resources to leverage economies of scale
- Elastic; that is, they respond dynamically to changes in workload
- Easy to deploy and manage on demand because management of the underlying platform and infrastructure has been abstracted away
- Resilient against failure

> *"At our scale, we face the challenge of wanting our engineers to move very fast while also keeping our site up and running. In other words, we want 100% productivity, meaning everyone in product can create, change, and release features while also enabling 100% availability, meaning our site is completely available for our guests, hosts, and employees.... To accomplish this, we leverage service-level dashboards so teams can get a high-level overview of a service with both business and system metrics side by side."*
>
> ***Melanie Cebula, Software Engineer, Airbnb***

Why is cloud native important?

At the end of the day, cloud native is important because it enables organizations to achieve transformational digital and business outcomes by taking full advantage of the myriad benefits of the cloud.

A well-executed cloud native strategy lets you move faster and deliver greater value by:

- Improving customer experiences and fostering loyalty
- Creating competitive advantage through increased agility and faster time-to-market
- Disrupting the industry by enabling new business models
- Reducing costs through increased operating efficiency
- Driving higher profits by increasing customer value and creating new revenue streams

"Cloud native computing…deploys applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud native technologies enable software developers to build great products faster."

"Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds."[5]

3: "CNCF Survey: Use of Cloud Native Technologies in Production Has Grown Over 200%," Cloud Native Computing Foundation, August 2018
4: "The State of Cloud Native Security," Capsule8, Signal Sciences, and Duo, 2018
5: Cloud Native Computing Foundation

CHAPTER 3

Becoming Cloud Native: It Doesn’t Happen Overnight

Chapter 3: Becoming Cloud Native: It Doesn't Happen Overnight

Cloud native is more than migrating existing applications to the cloud or flipping a switch to turn on new cloud services. It's also more than developing your applications from scratch in the cloud. It's about how you develop and run applications, not just where those applications are deployed.

Which is why becoming cloud native doesn't happen the instant you leverage cloud services or infrastructure. Rather, it requires fundamental changes to processes, architecture, culture, and measurement. These changes take time and commitment, and the cycle is iterative and constant.

Three core capabilities for cloud native

Evolving all the moving parts of your software development life cycle to leverage the benefits of the cloud is a continuous, iterative process. To be successful along the way, there are three core capabilities that any organization must master: visibility, experimentation, and optimization. These capabilities help you overcome three major challenges in modern cloud environments:

- **Transformation is continuous:** Constantly emerging new technologies and approaches mean the job is never done.
- **Modern environments are complex:** There's an ever-growing number of components across the stack and it's difficult to keep track, especially when things go wrong.
- **Scaling is difficult:** Early success can be hard to grow and sustain, and it's challenging to show measurable business impact within a reasonable and predictable budget.

Best practice tip: Start small, then go big

Many of our customers have found that starting with a well-defined use case, application area, or business unit allows them to gain experience in cloud native approaches and quickly demonstrate success, which grows support throughout the rest of the organization. Based on your initial success, consider starting a cloud center of excellence to help the rest of the organization with its understanding and embodiment of cloud native.

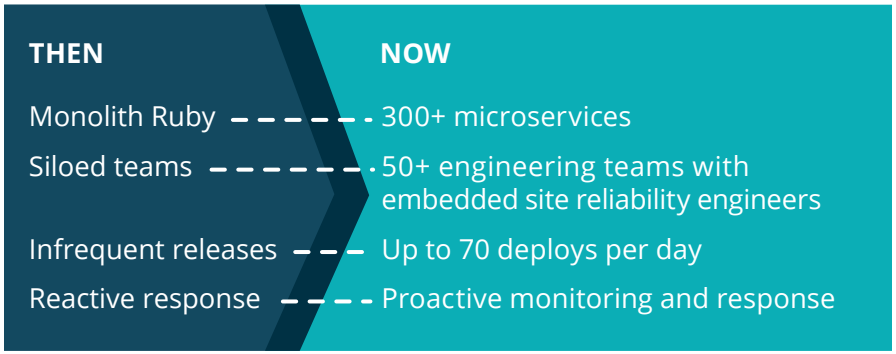
CHALLENGE	CORE CAPABILITY	IT IMPACT	BUSINESS IMPACT
Transformation is continuous	Visibility	Develop the insight and deep understanding to: • Effectively evaluate and integrate modern cloud technologies • Determine which of your applications are the best candidates for leveraging a cloud native approach • Decouple/break apart monolithic applications into microservices components	Visibility into the current environment gives you a better understanding of: • The current impact of application performance on the business • Which applications offer the greatest opportunity for improving business outcomes • Dependencies between different parts of your application and infrastructure and how they impact end users • How to measure the IT and business goals of moving to cloud native
Modern environments are complex	Experimentation	Leverage metrics, rapid feedback, and detailed analysis to: • Promote a culture of experimentation and innovation • Fail fast and fix faster in a complex, distributed system • Create a closed feedback loop for DevOps teams to thrive in complexity	A data-driven approach helps your teams: • Quickly find and resolve issues before, during, and after every deployment in order to optimize end-user experience • Understand the impact of new software and features on the customer experience and business outcomes • Align development resources with business goals
Scaling is difficult	Optimization	Improve scalability and performance of distributed systems by: • Automating and managing your distributed, containerized environment • Maximizing application reliability and scalability for a seamless customer experience even during usage spikes • Focusing budget and resources on efforts that drive business value	Optimizing the cloud environment helps you: • Meet business objectives quickly and cost effectively • Improve the customer experience • Justify and predict IT costs • Outperform the competition

Key success factor: Cloud native measurement

The need for measurement underpins the success of optimizing a modern environment. Context is critical in these complex, distributed cloud native environments. Without a data-driven approach, it's impossible to measure your maturity and results within any of the three capabilities.

And not only are measurement and context critical, but measurement best practices need to evolve to meet the demands of these modern systems. In a cloud native environment, traditional monitoring approaches tend to break in three critical ways:

- **Lack of visibility across complex systems:** When you adopt new technologies and services, you lose end-to-end monitoring capabilities and context across complex environments. Often, teams resort to manually correlating mountains of data across disparate tools to try to understand anomalies and dependencies.
- **Poor insight into contributing factors:** As environment complexity and experiment velocity increase, mean time to resolution (MTTR) tends to increase as well, because it becomes more difficult to pinpoint what went wrong across different levels of a distributed stack.



New Relic Then and Now: Our Iterative Journey to Cloud Native and DevOps

Best practice tip: Hire or grow platform skills

With the shift to modern technologies and infrastructures, a new role is emerging. Sometimes called platform engineers, these are the people responsible for building and evolving an enterprise's cloud native platform. From a skills perspective, someone in this role generally needs a strong background in infrastructure software and systems administration. Whether you ultimately create such a role or not, you should consider who on your team can help guide your organization to deploy the right cloud tools and infrastructure.

- **Difficulty scaling monitoring best practices across systems and teams:** Technology and process wins experienced in silos are difficult to convert to organization-wide best practices. Scaling success with measurable and sustainable business impact at a reasonable cost becomes challenging.

That's why modern environments require modern measurement.

> *"We've found that everything in reliability stems from our ability to measure what's going on. That means that in order to improve any process or the reliability of any system, we need to first measure how it's performing in production. In essence, monitoring becomes the first part of any reliability project that we undertake."*
>
> ***Karthik Nilakant, SRE Team Lead, Xero***

CHAPTER 4

Moving to Microservices

Chapter 4: Moving to Microservices

A central tenet of a cloud native approach is the concept of leveraging distributed systems. This includes either transforming monolithic applications into decomposed services or developing distributed systems from scratch. Either way, the idea is to enable continuous delivery and deployment of large, complex applications.

A microservices architecture is the approach of choice for breaking down applications into their smallest reasonable services. The main idea behind a microservice architecture is that applications are simpler to build, modify, and maintain when broken down into smaller pieces that work seamlessly together.

Because each microservice runs in its own container, service, or function, developers can build, manage, and scale each service independently. When implemented well, a microservices architecture has the potential to:

- Accelerate the velocity and increase the scale of software deployment
- Improve application scalability and optimize scaling decisions
- Increase business agility
- Improve fault isolation
- Enable smaller and more agile teams

From monolith to microservices

Identifying a monolithic application to break apart into more basic microservices components is not a trivial exercise. You need to collect quantitative and qualitative data to get context into which applications—and, more important, which components of those applications—are prime candidates for moving to microservices.

Understanding service meshes

As applications and the environments that host them become ever more distributed, a key challenge is maintaining a network between all the different services. Service meshes help with this by providing a software-defined networking layer that allows services to communicate with one another and supports service discovery, load balancing, and authentication capabilities. At the same time, service meshes add their own layer of complexity to a microservices environment. One way to gain visibility and understanding of this complex environment is with distributed tracing, which lets cloud native monitoring solutions instrument, propagate context, record, and visualize requests throughout distributed systems.

Once applications are decoupled, you'll need insight into which newly created microservices still have too many interdependencies with other applications and services. You will also need to analyze specific metrics to gauge the efficacy and results of your migration efforts. And like all things cloud native, this is an iterative process as your bottleneck has likely moved. You'll need to collect and analyze additional data to identify the next optimization candidate and repeat the process.

> *"Monitoring practices must evolve as you adopt microservices. You might see your service behaving badly, and it's not your code, but a downstream or upstream service that is affecting it. Failures will occur and you need your services to recover quickly; lean on improved monitoring standards to achieve rapid recovery."*
>
> ***Melanie Cebula, Software Engineer, Airbnb***

You can start by gathering key metrics such as transaction call volume and response time, which point to components of the monolithic application that are possibly degrading application performance and impacting the user experience. These are prime candidates for breaking down into microservices. It is important to also consult with colleagues who have written, designed, and/or maintained the application in order to supplement the context that your baseline data provides. Their experience and recommendations will help to guide the decomposition effort.

Key success factor: Simplify the complexity with visibility

For all the advantages of a microservices architecture, there are some downsides:

- Microservice architectures are complex
- Dependencies can be difficult to track
- Proper sizing is critical and not easy
- Successful implementation requires thoughtful and timely communication among DevOps teams

Data lets you optimize the benefits and minimize the disadvantages of a microservices architecture, which is why we start with visibility as a core capability for a cloud native journey. To achieve the technical and business goals that visibility makes possible, you need a complete view of your application performance—both as a monolith and decoupled into microservices—as well as the infrastructure that supports your application.

Cloud native, intelligent monitoring solutions can track the interconnecting dependencies between microservices so you can monitor the performance, operational characteristics, and service level objectives (SLOs) of those connections and be alerted when problems are detected.

Best practice tip: Use metrics to optimize microservices

After breaking your application into microservices, you can use metrics to identify microservices that are not yet completely independent. For instance, check to see if any microservices have a large volume of back-and-forth requests to the same downstream service. If so, that's a sign the microservice is not yet fully decoupled. Throughput is also an important indicator. If the number of calls per minute for a given microservice is significantly higher than the throughput of the overall application, that's a clear sign that the service is not decoupled.

CHAPTER 5

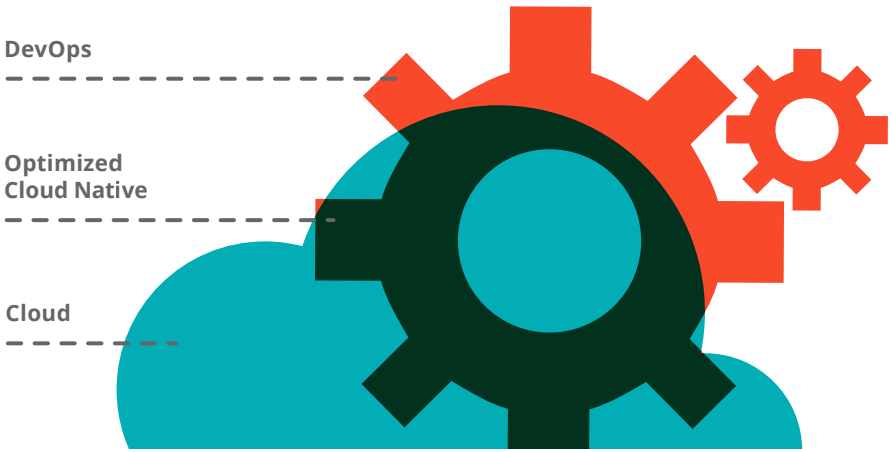
Embracing Cloud Native DevOps

Chapter 5: Embracing Cloud Native DevOps

Why are we talking about DevOps in an ebook about cloud native? Because cloud native—including the tools, technologies, and application architectures it entails—requires changes to your IT culture, organization, and processes.

The New Stack offers a handy definition of cloud native DevOps as "an emerging set of principles in the DevOps tools community that describes how people work together to build, deploy, and manage applications in a cloud native approach."[6]

Whether you have already adopted DevOps principles or are just getting started, you must be prepared to evolve your practices, roles, tools, and workflows to be successful in a modern environment. That's because cloud native infrastructure, architectures, and deployment methods are changing how people work—even those already using DevOps processes.



As organizations achieve increased technological and cultural maturity, the intersection between cloud and DevOps should grow.

Failing fast and recovering quickly

A critical aspect of a cloud native DevOps approach is fostering a culture of experimentation. Exploiting modern architectures to their full potential enables teams to build, test, and deploy software with the agility and speed of internet scale. Promoting and encouraging a culture of experimentation (and the failures that come along with it) maximizes the value of the cloud to your business by enabling:

- Faster time to market with new features and experiences
- Insight-driven innovation that challenges competitors
- Faster adoption of new technologies
- An organizational culture that encourages bold ideas
- Tighter alignment with business goals and outcomes

That said, there are some speed bumps to watch out for:

- Developer fatigue and unbalanced call rotations
- Communication and collaboration hiccups and growing pains
- Lack of contextual, intelligent insight into performance, usage, and process changes

Key success factor: Correlated data

The fuel for experimentation is data. And not just any and all data, but *correlated* data. This means that instead of parsing through the terabytes of data that your distributed systems generate, you have a curated view that gives you comprehensive insight and context into the "so what?" of every experiment, deployment, and release.

Correlated data, and the targeted actions it drives, help teams measure and track cloud native DevOps performance and optimize software delivery and business results in a modern environment. Most important, it gives teams the insight they need to nurture a culture of experimentation. Data removes emotion and finger-pointing and creates a common language across skills, experience, and roles.

To learn more about using DevOps in the real world, check out DevOps Done Right.

Best practice tip: Watch these metrics for DevOps success

DevOps teams are focused on the speed of development, delivery, and response to issues that occur in production. Consider tracking and establishing goals around these metrics:

- Lead time for changes
- Frequency of code releases
- Mean time to resolution (MTTR)

6: "Guide to Cloud Native DevOps," The New Stack, 2019.

CHAPTER 6

Optimizing Cloud Environments

Chapter 6: Optimizing Cloud Environments

Modern cloud environments are complex and constantly changing. Being cloud native isn't a discrete "A to B" journey that follows a simple formula. Rather, it's an iterative process that requires a continuous optimization loop to enable more efficient and effective usage of cloud resources while driving further gains in delivery speed, scalability, flexibility, and resilience.

Organizations typically rely on automation and orchestration as well as instrumentation and visibility to continuously optimize their modern environments.

Automation and orchestration

Automating a cloud native environment generally addresses the provisioning, configuration, and management of the cloud infrastructure. Orchestration is executing the tasks you've automated as part of a workflow. Think of it as organizing the flow of automation throughout your environment. Many tools and platforms offer a combination of automation and orchestration. Some solutions are provided by cloud vendors, some from independent software vendors, and others are open source.

As the scope of microservices and distributed systems increases in your cloud environment, you'll need a way to optimize managing them. While there are different and evolving approaches to optimizing and orchestrating distributed systems, two popular ones right now are:

- **Containers and container orchestration systems:** Cloud native applications architected using microservices often rely on containers to package the application's libraries and deployment processes. Although an application within a container operates independently from those in other containers, it still answers to directives from the kernel or other orchestration tool. Such orchestration tools are essential to manage large numbers of containers. Among the various alternatives, Kubernetes has become the container orchestration platform of choice among many organizations and cloud service providers.
- **Serverless services or functions:** Serverless is a general term that includes all cloud services that don't require you to spin up a dedicated server to use them. Serverless microservices or functions (extremely granular microservices that do just one thing) are spun up only when they are needed and you pay only for the resources used when they are running. For functions or microservices that are not constantly used, serverless enables scalability and flexibility while delivering real cost savings.

Best practice tip: Follow these practices for your Kubernetes environment

Before you build your first containers and deploy applications in your Kubernetes environment, keep these best practices in mind:

- Keep your container images small to improve time-to-build and time-to-pull
- Use resource requests and limits to avoid cluster misuse
- Take advantage of namespaces to improve manageability, security, and performance

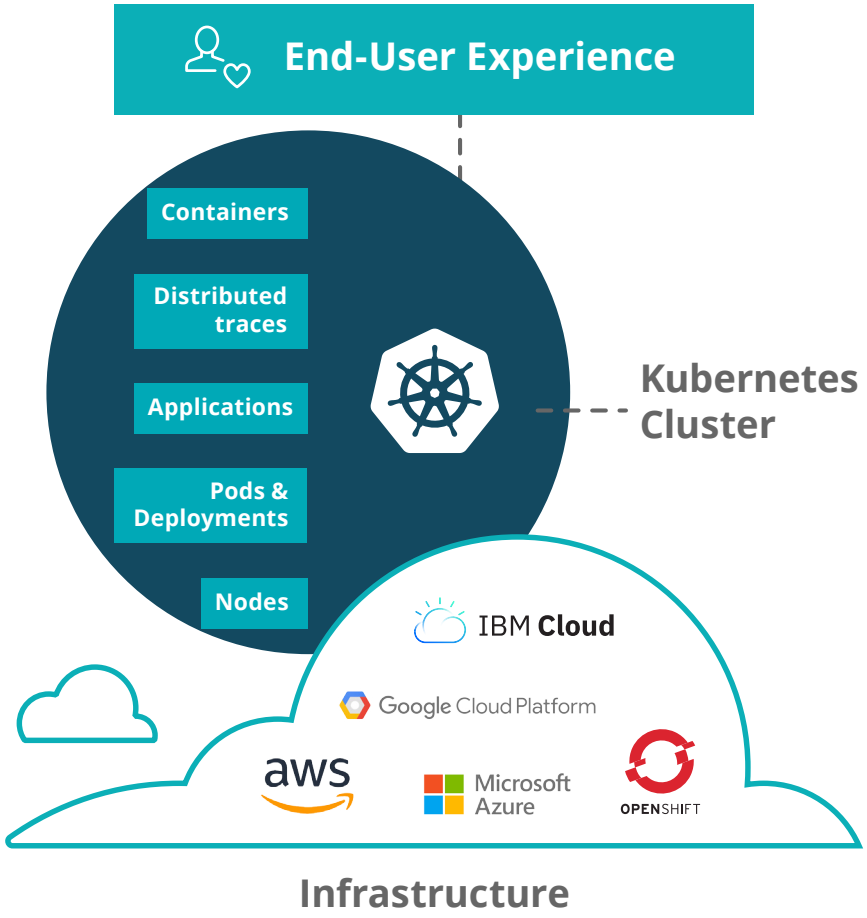
Why use container orchestration systems or serverless functions in your cloud native approach? Because they:

- Enable horizontal scaling
- Help teams move faster
- Optimize the use of cloud resources for greater cost efficiency

Instrumentation and visibility

As cloud native systems dynamically scale, maintaining control and visibility into resources used and the impact on costs and budgets becomes increasingly problematic. Once again, modern monitoring plays a critical role. You need to proactively monitor data and metadata for nodes, deployments, pods, and containers, as well as frontend and backend applications, distributed traces, and hosts running in your Kubernetes clusters.

You also need to look regularly and closely at how your applications and services are architected and utilized. This data-oriented approach allows you to rightsize your instances, fine-tune your databases, modify your storage usage, better configure your load balancers, and even re-architect your applications. This kind of thinking about your cloud utilization and spend will help you optimize your environment and justify the cost.



CHAPTER 7

Closing the Feedback Loop

Chapter 7: Closing the Feedback Loop

By leveraging cloud native technologies and approaches, companies have the ability to deliver software updates faster than ever—up to thousands of times a day. However, success depends on tighter, faster feedback loops about bugs, new features, performance, customer experience, and targeted business outcomes.

Author, researcher, and DevOps expert Gene Kim names amplified feedback loops as one of the "Three Ways" that frame the processes, procedures, and practices of DevOps. He cites the importance of creating right-to-left feedback loops, with the goal of shortening and amplifying them so corrections can be made continually.

To that end, you should be tracking key performance indicators (KPIs) related to the three core capabilities for cloud native so teams have a common frame of reference with which to analyze and improve the system:

CORE CAPABILITY	SAMPLE KPIS
Visibility	Availability, response time, Apdex, error rate, throughput, cloud cost
Experimentation	Number of deploys, lead time for changes, mean time to resolution (MTTR)
Optimization	Cloud cost, errors by pod, container memory usage, number of missing pods, response time by customer segment

Focus on outcomes

Taking feedback loops a step further, the Mobius Framework uses measurements to provide insight into outcomes that enable DevOps teams to focus on the business results they need to deliver. This ties back to our earlier discussion about showing how infrastructure and application changes ultimately impact the customer experience and business outcomes. If the results can't be measurably felt by the business, it's hard to argue the initiative was successful.

While increased velocity is one goal of enabling a cloud native environment, that velocity can be useless or even counterproductive without the right data to direct that velocity to the right problem with measurable outcomes for everyone to see.

Key success factor: Intelligence

Modern cloud environments can generate a substantial amount of "noise"; that is, enormous volumes of useless data from every nook and cranny of your distributed systems. When you instrument from end to end in your cloud environment, it becomes impossible for humans alone to keep up with the sheer volume of data generated—let alone contextualize and analyze it in a way that is meaningful in the feedback loop.

That's why cloud native organizations need a cloud native monitoring solution that manages and correlates all the data to help teams analyze and understand what matters and relate that back to outcomes. That solution should ideally provide:

- **Expanded visibility** across the environment to simplify the complexity
- **Analytics and intelligence** sourced from queried events, distributed traces, and aggregate data to provide the context teams need to quickly troubleshoot and optimize
- **Prescriptive solutions and support** that give you expertise and guidance for automating and optimizing distributed environments

Next Steps

While adopting a cloud native approach may not always be fast or easy, the rewards can make the effort worthwhile for both IT and the business.

As you start or progress further on your cloud native journey, ask yourself these four questions to assess where your organization is relative to the three core competencies we've discussed:

1. Do we have end-to-end visibility?
2. Can our engineers connect the dots between infrastructure changes, application deployments, and end-user experience?
3. Do we know what steps we need to take to improve the reliability and scalability of our systems?
4. Do we have a clear set of objectives for reaching the next level of cloud maturity?

Ready to embrace the cloud native era?

Get started at newrelic.com/solutions/cloud-native

> *"The health of our platform ties back to customer experience. We need to understand if systems are overloaded or if changes that we've made are causing a negative impact. Given the interconnectedness of the system, being able to bring monitoring metrics from infrastructure, application, and customer experience all into one place helps us to isolate issues more easily."*
>
> ***Karthik Nilakant, Site Reliability Engineer, Xero***