Cloud vs. Cloud-Native

Understanding the Cloud-Native Paradigm Shift

Authored by Jason Bloomberg, President at Intellyx

Despite its buzzword status, cloud-native computing goes well beyond cloud computing to redefine how organizations deliver and leverage dynamic software at scale.

While the open-source container orchestrator Kubernetes is at the epicenter of the cloud-native movement, cloud-native computing takes cloud best practices and extends them to the entire IT landscape, well beyond the scope of Kubernetes.

So many aspects of business technology transform when organizations move to cloud-native, in fact, that this new approach is a paradigm shift in the true sense of the word, as it will change everything about the way organizations create, deploy, and leverage software assets at scale.

The Context for Cloud-Native

It was a mere generation ago that an application was simply a computer program – a chuck of source code you'd compile into an executable for running on a single computer.

Then along came DLLs. And JavaBeans and Web Services. And now microservices. What was once a single binary now consists of multiple disparate bits of code, running in virtual machines or containers or even made up of nothing but serverless functions – in the cloud or on-premises or both.

To make matters worse, these application elements are even more ephemeral than before, coming and going on the order of fractions of a second to meet ever-changing scalability requirements.

Creating such applications is unquestionably more difficult than writing programs in the monolithic app days. Running them – especially in scaled out, enterprise environments – is orders of magnitude more complex as well.

“

Fundamental requirement for better and faster software is at the heart of cloud-native computing.

Why go to so much trouble? Why do we need to endure such onerous complexity?

The answer: the business – and our customers – demand new capabilities and a new level of experience that require a paradigm shift in how software meets business needs.

We're in the digital era, where software is central to most organizations' value propositions to their customer base. Not only does software form the backbone of the modern organization, it infuses all interactions within a company and among its customers.

What one company (or public sector organization) does, so too must its competitors. Fast beats slow. Dynamic beats static. Customers are fickle. They always want more and better.

And delivering on this value – better than the competition can – depends upon agile, fast-moving, dynamic software. This fundamental requirement for better and faster software is at the heart of cloud-native computing.

What is ‘Cloud Native’?

The first thing to understand about ***cloud-native*** computing is that it is quite different from ***cloud*** computing. Cloud computing represented a shift in the way organizations provisioned, managed and paid for compute, network, and storage resources – but as the truism says, the cloud is really nothing more than someone else’s computer.

Cloud-native computing means bringing cloud-centric best practices to all software and IT generally, whether that be in the cloud or on-premises – or both. Cloud-native extends the core best practices of the cloud to all of IT.

Cloud-native computing includes everything special about the cloud, and then adds many other fundamental capabilities, including how to support ephemeral, elastic software infrastructure at scale. However, cloud-native doesn’t actually require the cloud. In fact, it’s possible to be cloud-native and not be on the cloud at all, and vice versa.

The key to implementing a cloud-native strategy is to apply coherent management abstractions across the entire hybrid IT environment, comprising whatever environments are essential to the organization – public cloud, private cloud, on-premises virtualized environments, and legacy.

Abstractions provide essential simplicity and usability while masking the complexity underneath. To implement the abstractions necessary for a comprehensive cloud-native approach, it’s essential to get a firm handle on such complexity.

One evolving technology trend has become instrumental for dealing with this inherent complexity underlying cloud-native abstractions: ***containers***.

At their most basic level, containers are essentially a next-generation approach to virtualization. However, instead of the familiar virtual machines (VMs) that may take several minutes to spin up, containers may only take milliseconds to spin up – and to spin down.

“

Cloud-native computing means bringing cloud-centric best practices to all software and IT generally, whether that be in the cloud or on-premises – or both.

Containers are inherently ***ephemeral***. Like old postcards from people, you can't remember and ticket stubs from games long since played, ephemera are objects that no one ever intended to keep around for very long. Just so with containers.

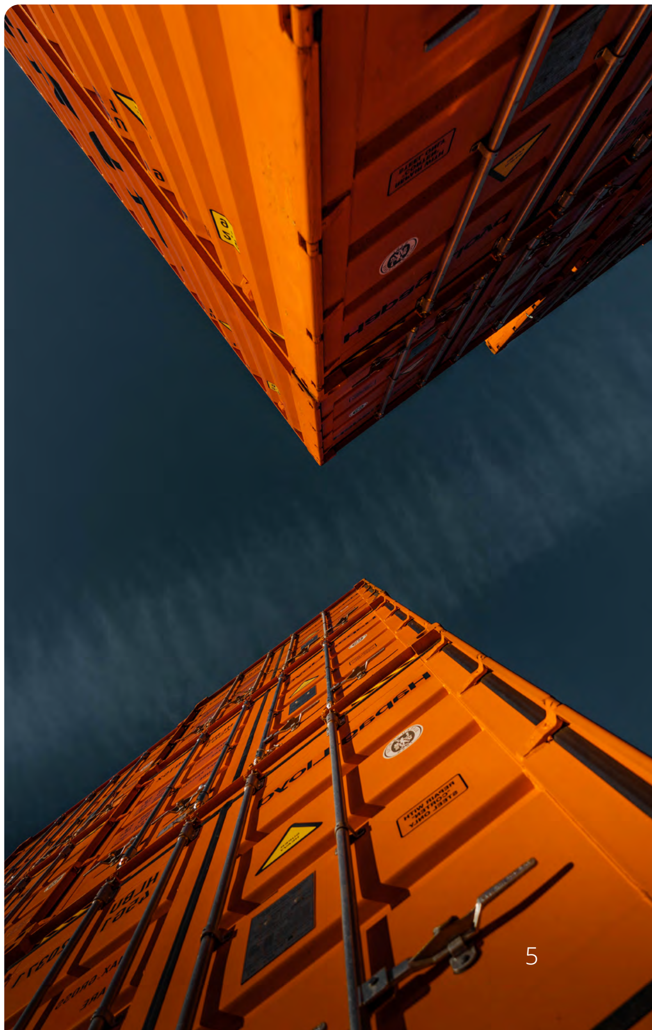
Managing this ephemerality in the context of delivering cloud-native application functionality requires a container orchestration platform, which is why the epicenter of the cloud-native movement is *Kubernetes*.

Kubernetes is container orchestration software – essentially, plumbing for running enterprise-class software in the cloud. Kubernetes has practically exploded onto the enterprise infrastructure scene, with a number of open source and vendor-led 'flavors' of Kubernetes in the market.

Red Hat offers OpenShift. Google delivers Google Kubernetes Engine (GKE) and Anthos, its cloud services platform. Then there's PKS (Pivotal), EKS (Amazon), and AKS (Microsoft), as well as several others.

Regardless of the particular flavor, as the leading platform technology underlying containers, Kubernetes is at the heart of how both enterprises and Web-scale companies will run their technology for years to come.

Kubernetes is container orchestration software – essentially, plumbing for running enterprise-class software in the cloud.



Beyond Kubernetes

There is more to cloud-native computing than simply leveraging Kubernetes, however. Cloud-native infrastructure includes a mix of VMs, containers, and serverless functions as appropriate to meet the business need.

In fact, cloud-native is more about abstracting complex hybrid IT infrastructures to provide a seamless management and deployment experience for both operators and developers. This broad vision for cloud-native goes well beyond what Kubernetes alone can provide.

The entire cloud-native computing landscape – Kubernetes and beyond – follows an 'infrastructure as code' approach, only in this case, the 'code' in question represents declarative configuration metadata.

In other words, operators create metadata representations of the behavior of the infrastructure as well as the applications that run on them. The infrastructure then implements those representations automatically.

You can think of these metadata representations as business policies that distill the business intent for the applications. A change in business intent will result in a change in policy, which will change the behavior of the underlying technology.

Simply putting apps in Kubernetes is not the same thing as becoming cloud-native. To achieve the full benefit of cloud-native computing, following the full breadth of cloud-native computing principles is essential.

The good news: the transition to cloud-native computing is not a sudden leap. It can easily be a gradual process, where short-term business drivers frame its pace and phasing.

The Cloud-Native Paradigm Shift

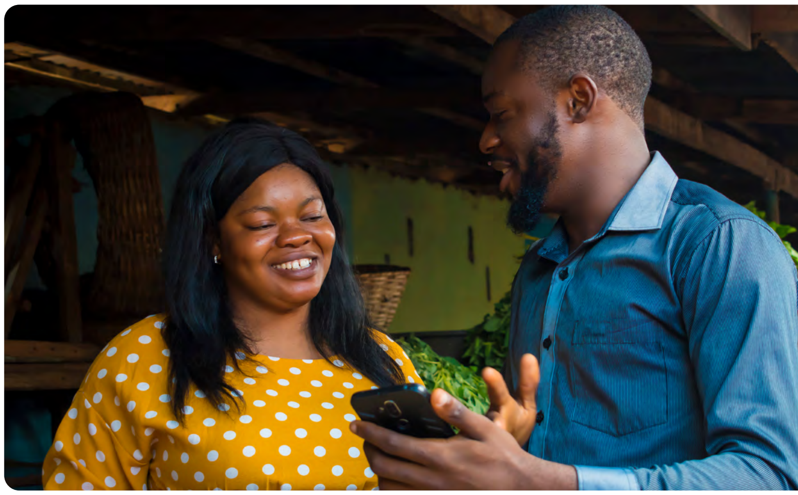
Cloud-native computing is more than an approach to implementing software infrastructure at scale. In reality, it is a paradigm shift that changes everything we know about how to code, engineer, integrate, architect, run, and secure modern applications.

What, then, is a 'paradigm shift'? We've all heard the term, but in many cases, people bandy it about haphazardly, relegating it to the status of a marketing buzzword.

Not so in this case. The term dates to the 1962 book ***The Structure of Scientific Revolutions***. In this book, physicist and philosopher Thomas Kuhn coined the term 'paradigm shift' to mean a relatively abrupt transition between two bodies of beliefs, techniques, and values that a scientific community shares.

His focus in this book was on the notion of scientific revolutions, consisting of paradigm shifts between one scientific paradigm and another. Over the years, however, the idea of a paradigm shift expanded well beyond the cloistered realms of science.

Today, a paradigm shift is not simply a change in opinion or technique but an overall transition in how an entire community approaches a general problem space.



Paradigm shifts can be challenging to get one's head around because so many things change from the previous paradigm, seemingly all at once. Cloud-native computing is no different.

Cloud-native computing changes the approach to coding, software deployment, operations, integration, architecture, and security, among others. How we're tackling each of these priorities has fundamentally changed from our pre-cloud-native days.

Cloud-native computing also requires a new paradigm for governance, as organizations must be able to define and enforce regulatory and other

Cloud-native computing changes the approach to coding, software deployment, operations, integration, architecture, and security, among others

business policies without slowing down software deployment – a task that was essentially impossible before the cloud-native approach to governance.

How we understand and deal with hybrid IT also changes under the cloud-native paradigm. The previous paradigm treated hybrid IT as a heterogeneous mix of environments that led to increasingly complex management challenges.

In contrast, the cloud-native approach requires a comprehensive, end-to-end control plane that empowers the organization to manage a diverse, intentional portfolio of IT resources, even as that portfolio continues to evolve.

Shifting the Paradigm for Cloud-Native Applications

The fundamental nature of applications also transforms within this new paradigm – how to architect and build them, deploy them, and gain business value from them.

Cloud-native applications also reflect the complex diversity of technology contexts that cloud-native computing brings to the table. Such applications may have dynamic end-user elements and sophisticated microservices-based back-end components running on Kubernetes in one or more clouds.

Any application that supports end-to-end digital requirements while delivering the agility that enterprises require from their software would constitute examples of cloud-native applications.

Such applications, therefore, take full advantage of a distributed computing environment – but go well beyond the n-tier and service-oriented applications that characterize earlier application architecture paradigms.

Cloud-native apps are more likely to have ephemeral components while also depending on a combination of modern event-driven technologies as well as RESTful APIs.

It's no surprise, therefore, that cloud-native applications can be so complex. They have *more components* that are *more dynamic* and *interact in more ways* than previous generations of distributed computing applications.

Transitioning to Cloud-Native

There's no question that cloud-native computing is exceptionally complicated. Kubernetes itself is open-source and represents an ecosystem of open-source products that only the boldest of organizations would hazard to assemble and configure themselves.

Fortunately, dealing with all the open-source bits of code on a do-it-yourself basis is rarely the approach most companies take. To address this need, many firms offer various configurations of Kubernetes-as-a-Service (KaaS), where the vendor handles the underlying infrastructure and much of the configuration.

However, even these KaaS offerings can be difficult to implement in practice, as the organization must still architect and build applications that run in a cloud-native environment.

The cloud-native paradigm shift entirely reworks the best practices for architecting applications. Designing, deploying, and managing microservices requires a rethink of how software runs and interacts with other pieces of software in the environment.

Designing, deploying, and managing microservices requires a rethink of how software runs and interacts with other pieces of software in the environment.

The software lifecycle also transforms in a cloud-native world. Rapid development of dynamic applications at scale becomes the primary goal, requiring organizations to adopt modern software development best practices, including DevSecOps, CI/CD, GitOps, and to an increasing extent, low-code.

Low-code, in fact, offers several benefits for building cloud-native applications. It can accelerate the work of developers, preventing hand-coding from becoming the bottleneck that slows down deployment.

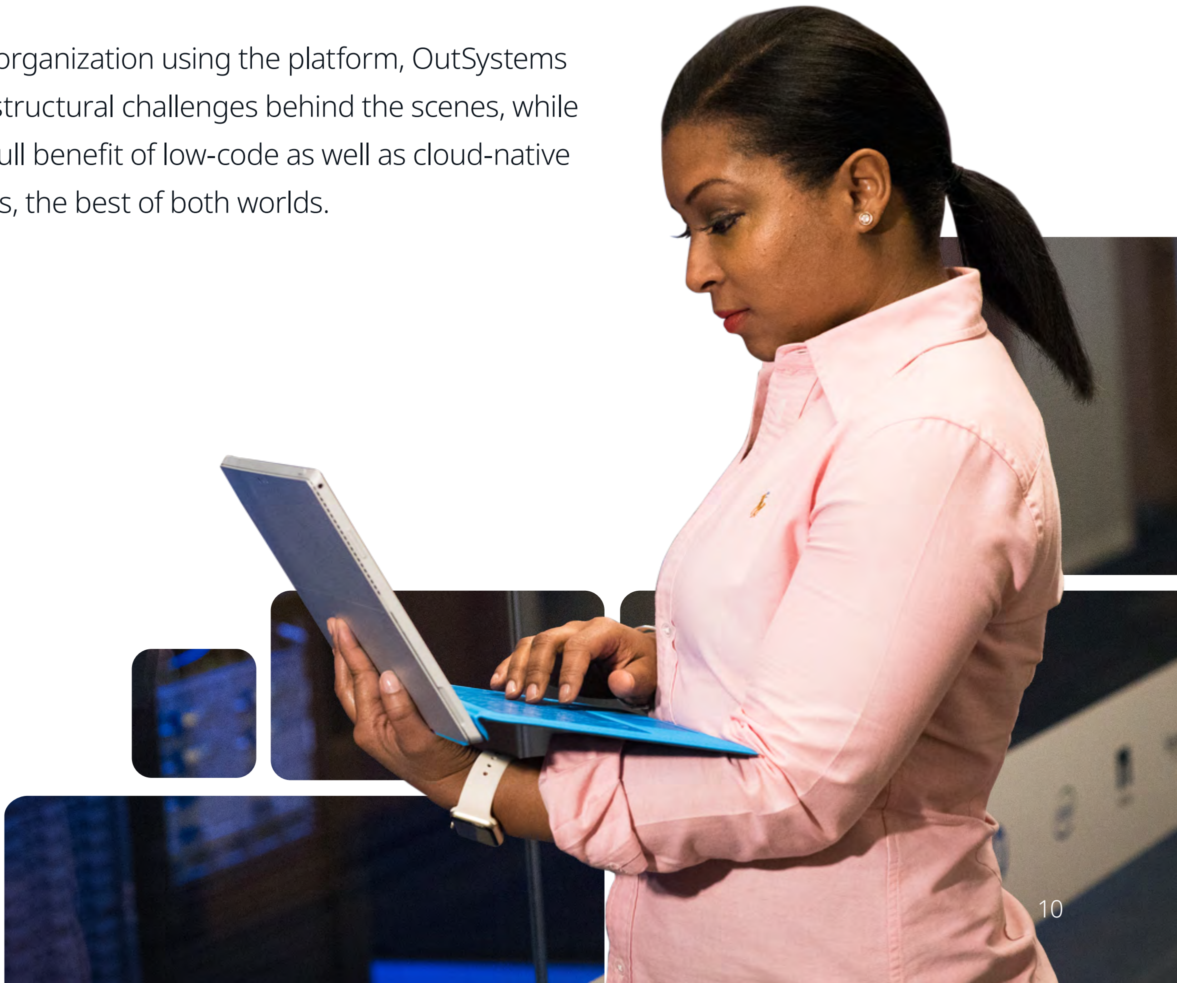
Low-code also fosters collaboration among different participants in the application development process, including developers, business stakeholders, designers, and analysts.

However, the most important benefit of low-code for building cloud-native applications is how it simplifies the process of updating microservices within the context of the overall application.

Given the complex interrelationships among microservices in the typical cloud-native app, a low-code approach can facilitate any required changes while maintaining the appropriate coordination across the entire application.

When a vendor like OutSystems provides a low-code cloud-native platform, the benefits of low-code combine with those of KaaS.

From the viewpoint of the organization using the platform, OutSystems handles the complex infrastructural challenges behind the scenes, while the organization gets the full benefit of low-code as well as cloud-native computing – in other words, the best of both worlds.



The Intellyx Take

As with any paradigm shift, there's no requirement that organizations adopt cloud-native computing in one fell swoop or that one company must adopt it the same way another does.

In reality, cloud-native computing consists of a plethora of individual, interrelated practices and technologies. Any cloud-native roadmap must take into account this complexity as it also considers the business priorities for the paradigm shift.

It's no surprise, therefore, that cloud-native adoption is proceeding somewhat chaotically. Some enterprises are all-in with the approach, while others are still dipping their collective toes in the water. Others still are waiting on the sidelines until the barriers to entry are sufficiently lowered.

Regardless of your organization's level of cloud-native maturity, there are steps you can take to reduce the risk of making such a big move while meeting business requirements along the way. Remember, you don't have to be perfect to win in the marketplace – you only have to be better than the competition.

www.outsystems.com