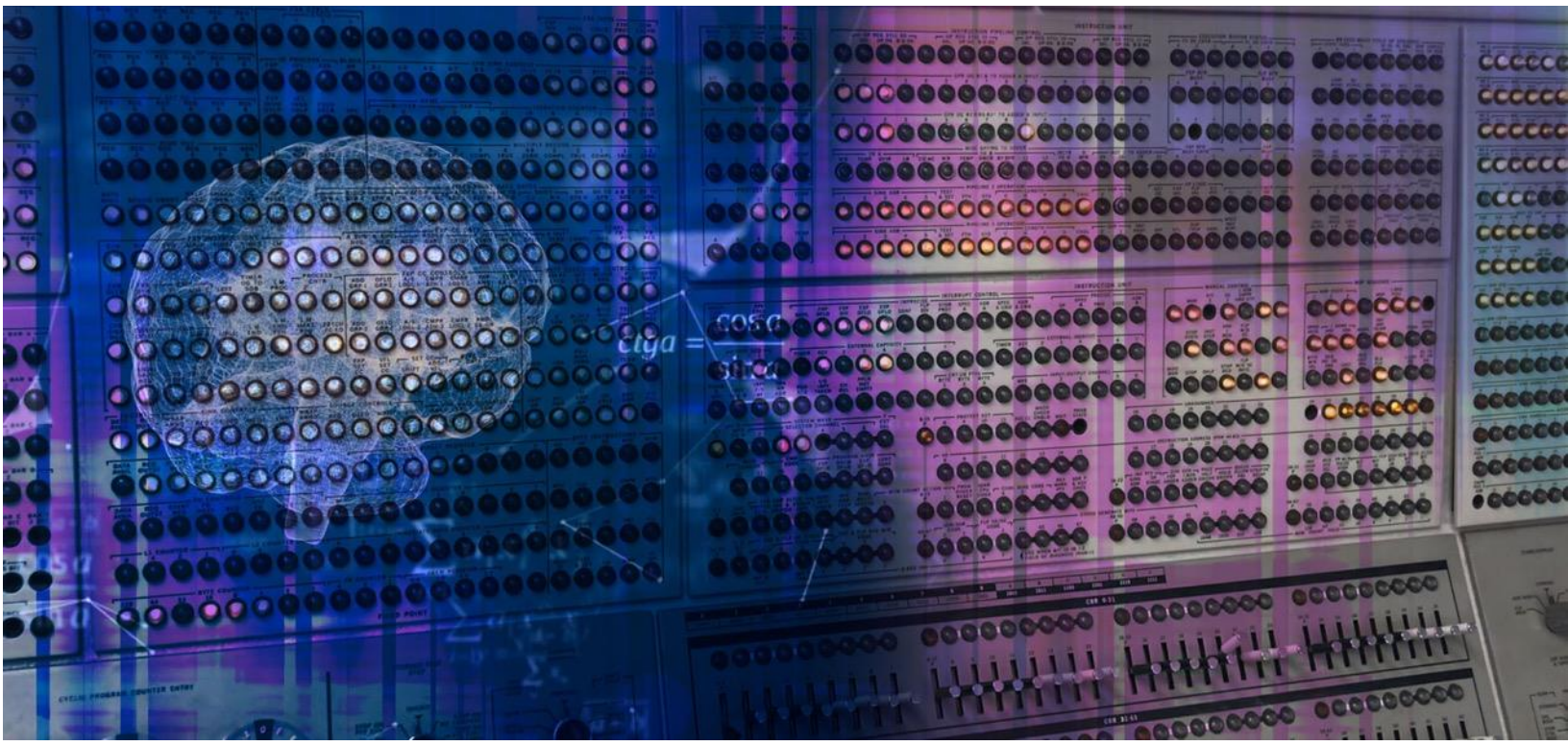


White Paper

How to Bring Cloud-Native Computing to the Enterprise

Jason Bloomberg
President, Intellyx

August 2019

As the complexity of modern IT continues to explode with no limits in sight, organizations are increasingly leveraging cloud-native computing to bring the best practices of the cloud to their entire IT organization.

An essential enabling technology for this next-generation approach to architecting IT infrastructure is the use of containers in conjunction with the Kubernetes container orchestration platform.

Containers bring dramatically improved flexibility to the applications that enterprises put in front of customers, but lack a sufficiently comprehensive way to maintain persistent information over time, because containers are inherently stateless.

Managing state information in such a stateless environment, therefore, is one of the primary challenges facing enterprise deployments of Kubernetes. Both, ROBIN platform and ROBIN storage, solve the challenges by enabling stateful workloads that follow cloud-native principles.

What is 'Cloud-Native'?

For all its transformative power and business value, cloud computing has unquestionably been a lightning rod for hype.

This buzzwordiness continues with little hope of abating. Today, it swirls around such terms as cloud-native. In common parlance, cloud-native refers to software that developers have built in – and for – the cloud.

This concept is strikingly important to how enterprises take advantage of the cloud – and even more so, extend the value of the cloud to their IT organizations at large.

From the enterprise perspective, this definition of cloud-native might apply to some of the new software they're building, but the cloud-native world would forever be separate from the on-premises context for enterprise IT that has been with us for generations.

Fortunately, this definition is shifting. Today, 'cloud-native' is more than 'cloud only.' It means bringing cloud-centric best practices to software and IT generally, whether that be in the cloud or on premises – or both.

'Cloud-native' is more than 'cloud only.' It means bringing cloud-centric best practices to software and IT generally, whether that be in the cloud or on premises – or both.

The devil, of course, is in the details – and details there are, in spades. The reality is that there are many moving parts to cloud-native architecture. And just as service-oriented architecture and virtualization-based cloud architecture built upon n-tier architecture, the cloud-native architecture also leverages the approaches that came before, while breaking new ground.

There's no arguing with the fact, however, that such re-architecting is hard. And you have to get the details right.

The Rise of Containers

The key to implementing a cloud-native architecture strategy is to apply coherent abstractions across the entire hybrid IT environment, comprising of whatever environments are important 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.

We typically think of containers as stateless: send them some information, let them do their thing and send the results on, without keeping track of anything. After all, a container may pop up or disappear at any time, so it probably wouldn't make sense to store anything important in one of them.

The next thing you have to understand about containers is that they 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.

As a result, we typically think of containers as stateless: send them some information, let them do their thing and send the results on, without keeping track of anything. After all, a container may pop up or disappear at any time, so it probably wouldn't make sense to store anything important in one of them.

Deciding where to maintain the state information that an application depends on, therefore, is a critical part of being cloud-native.

The Rise of Kubernetes

At the epicenter of the cloud-native movement is *Kubernetes* – and today, to understand the cloud-native approach, it's essential to understand Kubernetes.

Kubernetes is container orchestration software – essentially, plumbing for running enterprise-class software in the cloud. Containers, in fact, represent the reinvention of virtualization for the cloud. Virtualization was the core innovation that transformed enterprise IT in the last decade, and containers promise the same for the next.

Kubernetes has essentially 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.

Dealing with State in a Stateless Environment

Kubernetes, however, is inherently stateless, a necessary side-effect of containers' inherent ephemerality. After all, you wouldn't want to store data in one if it could disappear at a moment's notice.

Any enterprise looking to leverage Kubernetes, therefore, must somehow resolve the challenge of maintaining state in this essentially stateless environment. Kubernetes must handle state information – both persistent data in databases and file systems as well as more transient (but still persistent) application state in caches.

Organizations must also realize that it's unlikely that all of their code will consist of greenfield microservices in containers. They must also resolve issues of siloed infrastructure, as existing servers – both virtualized and unvirtualized – will continue to exist in an essentially hybrid environment. It is essential, therefore, that the operations team properly manage such environments to maximize their agility, portability, and efficiency. In the end, they will save money for their organizations.

Given the dynamic ephemerality of containers, simply connecting them to a database or file system as though the applications running in those containers were leveraging a traditional virtual machine environment would lead to a range of performance issues and the possibility of lost data in the event of unexpected failures.

To accomplish such state management in a stateless environment, Kubernetes follows cloud-native architectural best practice by abstracting storage via codeless principles and exposing such stateful resources via APIs.

To accomplish such state management in a stateless environment, Kubernetes follows cloud-native architectural best practice by abstracting storage via codeless principles and exposing such stateful resources via APIs. This approach allows for whatever availability and resilience the organization requires from its persistence tier without requiring the containers themselves to be stateful.

APIs, however, are only part of the story. Few vendors go beyond API support to address the cloud-native challenges that containers bring to the fore. One such vendor is ROBIN. "Containers might be ephemeral, but ROBIN storage is not," explains Partha Seetala, CTO of ROBIN. "ROBIN automatically and transparently (to the workload/application) switches volume mounts from one host to the other without requiring data copies."

In other words, because ROBIN abstracts the storage layer, it addresses both of the concerns with container-savvy databases: it can support the failover of the database without the need for copying data, and it also ensures that containers run on different hardware when necessary in order to guarantee resilience.

Stateful applications come in many forms. Database management systems, such as MySQL, PostgreSQL, and MongoDB are necessarily stateful, as they form the core of any application's persistence tier.

In addition, real-time data processing technologies like Apache Kafka and Apache Spark have stateful elements, although they too handle most of their operations in a stateless manner. Management technologies like Elastic and Splunk must also maintain state information, as do cache technologies like Memcached.

ROBIN offers cloud-native storage and data management for Kubernetes, and can manage state information at the application level, abstracting the storage tier entirely. This capability provides increased resilience for applications running on Kubernetes, including big data apps and databases running either in the cloud, on-premises, or any combination.

Modernizing Legacy Assets the Cloud-Native Way

Kubernetes provides cloud-native approaches to modernizing legacy assets. How enterprises modernize their tech is itself undergoing its own transformation – and Kubernetes is rapidly becoming an essential part of such transformations.

Containers are all very well and good, but to bring cloud-native architectures to the enterprise, IT leaders must to come up with a strategy for dealing with legacy IT assets. Modernization, therefore, is a central part of most enterprises' cloud-native strategies.

Over the years, one generation of technology innovation after another chiseled away at the modernization challenge. Cloud computing, and in turn, containers built on the advances of REST and virtualization, further cementing the layers of abstraction that enabled organizations to take an increasingly modular approach to modernization.

The result: modernization is no longer a two-sided dilemma, a high-risk choice between 'rip and replace' vs. 'leave and layer.' There are now many options to add to the mix.

In some cases, it's possible to modernize software in place – for example, on mainframes, as modern mainframe technologies enable organizations to continue to leverage the venerable hardware platform as they modernize its software that runs on it.

In other situations, it's possible to 'lift and shift' – moving software from less flexible platforms into the cloud, where in some cases, it can take advantage of the cloud-native architectural benefits of scalability, elasticity, and on-demand availability.

When it's possible to modularize legacy software, containers can provide an upgrade path that lowers the risk of all-or-nothing rip and replace strategies. However, making the decision of which option is best for a particular circumstance can be tricky. Having the right technology in place is essential.

How Cloud-Native Architectures Help with Legacy Modernization

Cloud-native architectures can maintain state properly by leveraging platforms like ROBIN's, but how do such architectures help with legacy modernization?

This paper is too short to provide an in-depth answer to this question, but it can discuss one example: placing an Oracle database cluster into Kubernetes, the most popular container orchestration platform.

For example, an Oracle shop might want to provide an Oracle as a 'database-as-a-service' offering that delivers the cloud-like rapid provisioning and horizontal scalability for its Oracle instances.

Oracle has long offered clustering capabilities for high availability via its Real Application Cluster (RAC) product. One might think, however, that putting an Oracle RAC cluster into a Kubernetes group, or 'pod' of containers would be sheer folly, as such pods are ephemeral.

Given the goal of such clustering is to guarantee high availability, would we really want such a cluster to be subject to the vicissitudes of ephemerality?

ROBIN solves this problem, because it deploys each instance of the Oracle RAC cluster in its own Kubernetes pod, while leveraging Oracle's cluster management to run within those pods. In this case, it is still necessary to provide clustered storage outside of Kubernetes and Oracle. ROBIN storage provides this capability.

Furthermore, when a server or pod fails, the infrastructure must relocate the pod from one server to another. Oracle has no idea how to manage such relocations. Even Kubernetes by itself can only relocate pods, but not storage. ROBIN enables both by dynamically switching storage and network bindings from the failed server to the healthy one.

Beyond Database Persistence

While this paper's discussion of state focuses on the database, there is actually more to this issue: *application state*.

Even in the n-tier days, we struggled with maintaining state on the presentation and application tiers, either through caches on such layers or via the heavily maligned browser cookie.

ROBIN is able to keep track of application state information, essentially preserving a seamless customer experience regardless of the appearance or disappearance of any pods or containers behind the scenes.

Kubernetes' essentially ephemeral nature ups this game. A pod might contain state information in the form of data, but also metadata associated with the application state – for example, what items are in a shopping cart, say, before the app has time to store such information in the database.

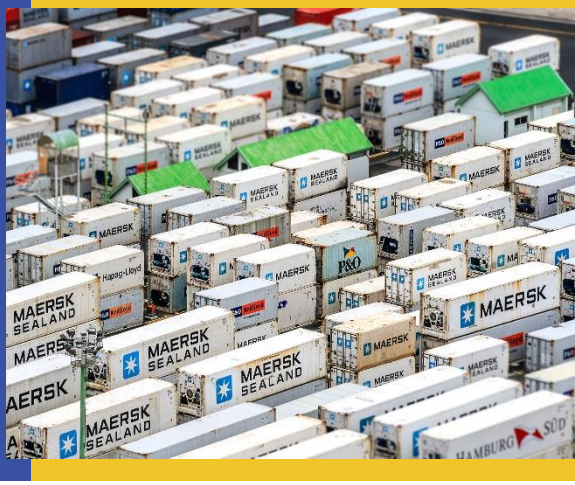
ROBIN is able to keep track of such application state information, essentially preserving a seamless customer experience regardless of the appearance or disappearance of any pods or containers behind the scenes.

And without such a customer experience, cloud-native architectures won't be able to rise to the challenges of the digital era.

The Intellyx Take: Putting the Cloud-Native Vision into Action

Containers may be an essential technology for implementing the simplifying abstraction of cloud-native computing – but make no mistake, this is no oversimplification. The reality of enterprise IT remains every bit as complicated as before. Perhaps even more so.

There are many moving parts – abstractions, persistence tiers, as well as applications and workloads – but the cloud-native approach gives us a path forward, enabling organizations to leverage increasingly powerful technologies to meet business needs.



With such increasing levels of complexity, however, arise concomitant improvements in how we deal with such complexity. True, there are many moving parts – abstractions, persistence tiers, as well as applications and workloads – but the cloud-native approach gives us a path forward, enabling organizations to leverage increasingly powerful technologies to meet business needs.

Cloud-native computing accepts this complexity as given. But for infrastructure teams who must ensure that everything works, at scale and at speed, they must leverage the best solutions available to the toughest infrastructure problems they face.

Managing state in an inherently stateless environment is just such a challenge. Without platforms like ROBIN, infrastructure teams will struggle to support the applications and workloads that provide value to customers in the digital era.

About the Author: Jason Bloomberg

Jason Bloomberg is a leading IT industry analyst, author, keynote speaker, and globally recognized expert on multiple disruptive trends in enterprise technology and digital transformation.

He is founder and president of Digital Transformation analyst firm Intellyx. He is ranked #5 on Onalytica's list of top Digital Transformation influencers for 2018 and #15 on Jax's list of top DevOps influencers for 2017, the only person to appear on both lists.

Mr. Bloomberg is the author or coauthor of four books, including *The Agile Architecture Revolution* (Wiley, 2013). His next book, *Low-Code for Dummies*, is due later this year.

About Robin.io

Robin.io brings advanced storage and data management that extend the Agility, Efficiency and Portability of Kubernetes to All Stateful Applications, even complex Big Data, Databases, AI/ML and Custom Apps, on Any Infrastructure, On-Premise, Hybrid Cloud or Multi-Cloud. ROBIN Platform is the industry's first hyper-converged solution that enables big data, databases and AI/ML as a Service on Kubernetes. ROBIN Storage delivers bare-metal performance and enables powerful data management capabilities such as snapshots, backup and migration to support even the most demanding data-intensive workloads. With a team that includes industry veterans from leading enterprise technology companies such as IBM, Microsoft, NetApp, Oracle, Red Hat and Veritas, Robin.io seeks to reinvent IT Infrastructure and put the focus back on what matters most – The Applications. The San Jose California-based company is backed by leading investors such as Clear Ventures, DN Capital, USAA and Hasso Plattner Ventures.

Website: www.robin.io