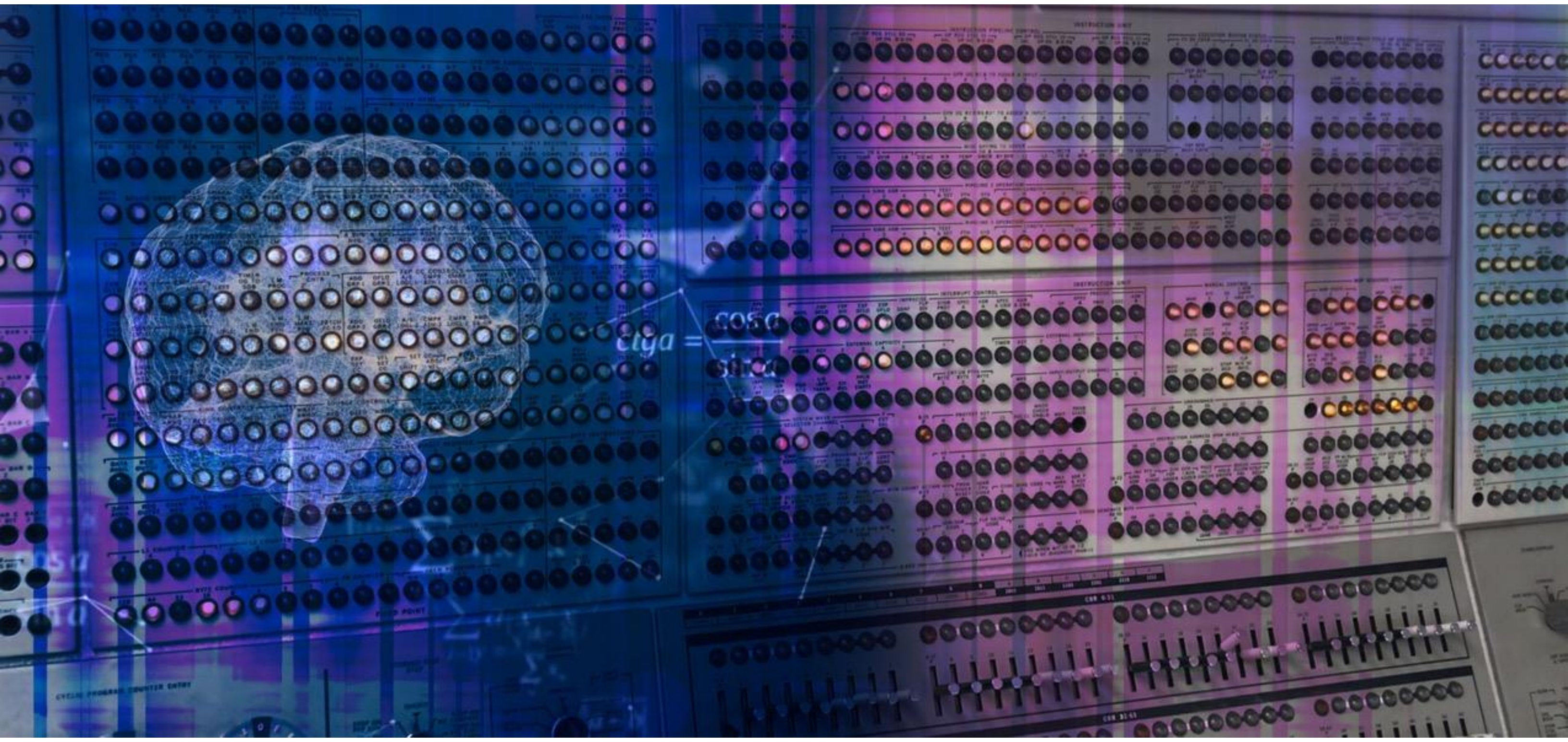


White Paper

Rethink the Modern Application

How to Think Like a Cloud-Native Engineer

Jason Bloomberg
President, Intellyx

July 2021

Since the early days of computing, the notion of an application has evolved several times. From a monolithic computer program to a distributed app and now to cloud-native applications, we've layered on abstractions and complexity in order to achieve unprecedented levels of scalability, resilience, and agility.

Cloud-native computing, however, is more than an evolutionary step. It represents a true paradigm shift that changes everything we know about how to code, engineer, integrate, architect, run, and secure modern applications.

The Transformed Application

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.

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.

Why go through 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.

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 the value propositions most organizations provide to their customer base. Not only does software form the backbone of the modern organization, it infuses all interactions that take place 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.

It's no wonder, therefore, that today's applications and the infrastructure they run on must rise to such complex challenges. It's no coincidence that microservices and containers have become the primary application delivery mechanisms in today's fast-moving enterprises.

If you've ever wondered why Kubernetes suddenly became the container orchestration standard, forming the foundation for the entirely new paradigm of cloud-native computing – look no further than this rapid, always-changing business context for modern software.

Cloud-native computing, however, means far more than simply using Kubernetes in a public cloud. Referring to it as a new paradigm is no exaggeration, as it requires different ways of thinking about the entire software landscape.

Cloud-native computing means far more than simply using Kubernetes in a public cloud. Referring to it as a new paradigm is no exaggeration, as it requires different ways of thinking about the entire software landscape.

Boiling down a paradigm shift to its basics is a tall order to be sure – but we need to start somewhere. Here, then, are five critical 'mind shifts' people need to undergo in order to gain an appreciation for the full import of cloud-native.

Mind Shift #1: Coding

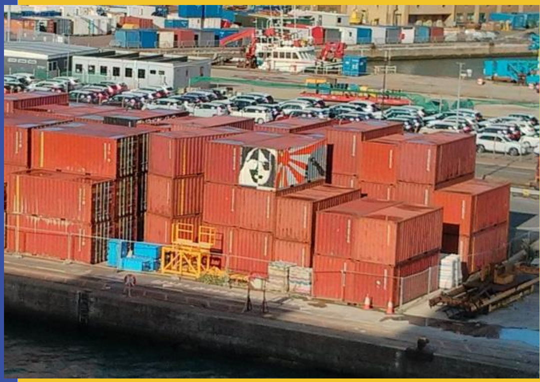
Since the 1960s, creating applications has meant writing lines of code. It simply boggles the mind that some of the largest programs in the world, from banks' core transaction processing applications to global airlines' reservations systems, were entirely hand-made, line by line – and many are still running to this day.

Over the years, many ingenious software vendors have rolled out software-writing tools of one sort or another, from CASE tools to 4GLs to RAD technologies. Today, the latest incarnations of this trend are low-code and no-code tools.

Software development professionals who have hand-coded for their entire careers may belittle such technologies, but the fact of the matter is that low-code/no-code is making inroads in the enterprise.

These tools help accelerate and streamline the work of professional developers, while empowering a broad range of less technical business users to create applications as well.

The fact of the matter is that low-code/no-code is making inroads in the enterprise. These tools help accelerate and streamline the work of professional developers, while empowering a broad range of less technical business users to create applications as well.



In the cloud-native world, anything that can simplify the software creation process is a welcome addition to the toolbox. Today's low-code/no-code tools, however, are generally not up to the task of crafting complex microservices-based applications that leverage the full power and flexibility of modern cloud-native technologies, but they are up to the task of stitching together apps from a number of APIs and adding a bit of extra glue to make them all work together.

Nevertheless, the interplay between low-code/no-code tools on the one hand and professional or 'pro-code' tools on the other continues unabated. Even the most rigorous of professional development tools can benefit from the visual simplifications that low-code can provide.

Coding for a cloud-native environment, therefore, leverages a combination of low-code, no-code, and pro-code tooling as appropriate for the task at hand, even as such tools continue to evolve.

Mind Shift #2: Software Deployment, Release, and Operations

There are still a surprising number of organizations that struggle with casting off the relics of waterfall thinking as they move to Agile approaches to building software.

In truth, however, Agile is itself over two decades old, and its flaws are showing: challenges with involving business stakeholders in the development process, a tendency toward dogmatism, and a tacit encouragement of homogeneous teams that lack sufficient diversity and inclusion.

In spite of this plethora of challenges, organizations are making progress. The intertwined trends of DevOps, CI/CD, and infrastructure as code are gradually overcoming the limitations of Agile, and any remnants of waterfall that remain.

DevOps brings a new collaborative ethos to the development process. CI/CD leverages increasingly sophisticated tooling to improve automation. And the 'cattle not pets' approach of infrastructure as code brings a new reliance on comprehensive declarative abstractions to even the most complex of software environments, ideally resulting in fully configuration-driven immutable infrastructure.

All of these trends – including the best of Agile – feed into what is now coalescing as cloud-native software engineering, as organizations extend software development best practices to software deployment and operations.

The new name for this approach is *GitOps*. GitOps is a cloud-native model for operations that takes into account the software engineering trends up to this point, including model-driven, configuration-based deployments onto immutable infrastructure that support dynamic production environments at scale.

GitOps extends the Git-oriented best practices of the software development world to ops, aligning with the configuration-based approach necessary for cloud-native operations – only now, the team uses Git to manage and deploy the configurations as well as source code.

On the other side of the GitOps coin: *shift-right deployments*. Not to be confused with nor replace the 'shift-left 'priorities of configuration-driven software engineering, shift-right reflects the fact that the production environment for cloud-native software is always in flux.

As such, ops teams must test deployed software in production via canary deployments, A/B testing, and other, more sophisticated applications of feature flagging.

GitOps provides the process controls and policy-based governance that shift-right approaches require without slowing down deployments – critical capabilities for cloud-native software engineering.

Mind Shift #3: Integration

In the mid-2000s, the rise of XML-based Web Services provided a level playing field for enterprise integration that no one had seen before. Instead of the hand-coded, point-to-point integrations in use at the time, Web Services promised standards-based interfaces that all software could agree upon.

The loose coupling that was supposed to be a central Web Services benefit, however, was difficult to achieve in practice. Even with the ESBs of the day, integration largely consisted of rigid, tightly-coupled connections. And in any case, ESBs weren't cloud-friendly.

By the end of the decade, REST had largely supplanted Web Services, bringing a lightweight, web-centric approach to integration. As REST matured, APIs reemerged – now loosely coupled, simple to implement and consume, and cloud-friendly. The API economy was born.

Today, while intuitive API development models have become increasingly essential to cloud-native development, APIs are nevertheless only a part of the cloud-native integration story.

In hybrid IT or B2B integration scenarios, REST-based APIs are still the primary tool for greasing the wheels of integration. Between microservices, however, service meshes are taking the lead.

In hybrid IT or B2B integration scenarios, REST-based APIs are still the primary tool for greasing the wheels of integration. Between microservices, however, service meshes are taking the lead.

Service meshes leverage configuration-based proxies to abstract microservice endpoints – essential when those endpoints are ephemeral. However, there is more to service meshes than proxy-based abstraction.

Many vendors are seeking to extend the power of the service mesh. Some companies are bringing the power of endpoint abstraction to data-centric operations in what we might call a data mesh. Also in the works: event meshes that are inherently asynchronous and event-driven.

This profusion of meshes, however, indicates many works in progress. It could certainly come to pass that these different flavors of meshes become simply different configurations of a core mesh capability.

Mind Shift #4: Application Architecture

The monolithic applications of the pre-Web days once ruled the data center, but today we've relegated them to the status of legacy.

The rise of distributed computing and the Web led to increasing demands for modular software and multiple tier architectures, and languages like Java and C# that excelled in supporting such applications.

Such architectures served their purpose to be sure – we could never have built the eBays or Yahoos of the world without them – but the scalability of such applications paled in comparison to the power of the cloud.

Cloud computing brought effectively unlimited horizontal scale to both born in the Web and eventually enterprise customers, along with a new focus on resilience over high availability that threw a wrench into the software architectures of the day.

Technologists now had to engineer programs correctly in order to achieve the cloud's promised benefits of scalability and resiliency. Microservices, containers, and eventually Kubernetes and serverless computing were the eventual result.

Software architects must intentionally group microservices into the appropriate domain contexts in order to maintain the coherence of the microservices architecture. Without such architectural grouping, microservices will quickly devolve into an 'anything will talk to anything' morass that will limit scalability and bring the network to its knees.

No longer did applications have to run in an execution environment. Instead, applications would bring their own execution environments along for the ride.

At the heart of this mind shift lies the nature of software modularization. Before cloud-native, modularization took place at the object level, as object-oriented programming provided the central organizing principle for modern software architectures.

In the cloud-native world, modularity takes place at the container and microservice level, or even at the function level in the case of serverless architectures. As a result, some of the familiar modularity patterns from OO apply, but others do not.

Most notably, in the absence of a common execution environment, microservices now require business domains that correspond to the business capabilities of the software – independent of any execution context for those microservices.

In other words, software architects must *intentionally* group microservices into the appropriate domain contexts in order to maintain the coherence of the microservices architecture. Without such architectural grouping, microservices will quickly devolve into an ‘anything will talk to anything ’morass that will limit scalability and bring the network to its knees.

The bottom line: cloud-native computing requires a new order of architectural maturity that builds upon and transforms the software architectures of the past, while adding new considerations that are essential for preserving the central advantages of cloud-native architectures.

Mind Shift #5: Security

You might think that it’s appropriate that we relegate security to our last mind shift. After all, security has often been an application development afterthought – and ensuring running software is adequately secure has long been someone else’s problem.

In the cloud-native world, we simply cannot afford such complacency. Not only is security everybody’s problem, but we must place security first on our list of priorities.

The ongoing maturation of application security techniques, as well as the rise of DevSecOps, illustrate the progress we’ve already made to achieve these security priorities. Developers are now the first line of defense for the software they build, and security is an integral part of the DevOps collaborative approach to software deployment.

However, there is more to the cloud-native security story: *cloud-native zero-trust*.

‘Zero trust’ is a security model that requires strict access controls, not trusting anything by default for any person, application, or service—even if it resides within a network perimeter.

Containers’ dynamic, ephemeral nature, as well as other cloud-native principles like declarative, configuration-driven behavior, require a top-to-bottom rethink of how zero-trust works.

Traditional zero trust centers on user permissions and IT resources with fixed network locations. However, in the cloud-native world, inherently dynamic, ephemeral resources make IP addresses useless for dealing with security challenges. The cloud-native context, therefore, must fully abstract all endpoints.

Furthermore, cloud-native zero-trust focuses on which endpoint is able to do what when—in other words, it keeps track of the activities that specific people, applications, devices, and services should be able to accomplish when they interact with any endpoint—and it blocks all other behaviors.

The Intellyx Take

The thing about paradigm shifts is that they end up changing everything. Cloud-native computing is no different.

You might think that cloud-native is all about containers and microservices. Or Kubernetes. Or service meshes. Or configuration-driven infrastructure as code. Or any of a number of other facets of what is truly a complex, difficult beast.

Always remember, however, why we’re bothering with cloud-native. Our businesses and our customers are demanding the agility, scale, and resilience that only cloud-native computing can deliver.

These stakeholders demand ongoing, unpredictable change from our software, and thus we must adopt change itself as a core competency in our software organizations and the enterprise at large.

There is more to the cloud-native story than five mind shifts can cover, of course. Stay tuned for the second paper in this series, where we extend the discussion to hybrid IT, edge computing – and beyond.

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 Thinkers360's Top 50 Global Thought Leaders and Influencers on Cloud Computing for 2020, among the top low-code analysts on the Influencer50 Low-Code50 Study for 2019, #5 on Onalytica's list of top Digital Transformation influencers for 2018, and #15 on Jax's list of top DevOps influencers for 2017.

Mr. Bloomberg is the author or coauthor of five books, including *Low-Code for Dummies*, published in October 2019.

About WSO2

Founded in 2005, WSO2 radically simplifies the way enterprises create, deliver, and scale digital experiences. Our cloud native, API-first approach helps developers and architects to innovate at speed and accelerate time to market. Customers choose us for our broad, integrated platform and our expertise in API management, enterprise integration, and identity and access management—the cornerstones of every successful digital transformation initiative. With offices in Australia, Brazil, Germany, Sri Lanka, the UAE, the UK, and the US, WSO2 employs over 800 engineers, consultants, and professionals worldwide. Today, hundreds of leading brands and thousands of global projects execute more than 18 trillion transactions annually using WSO2 technologies. Visit https://wso2.com to learn more. Follow WSO2 LinkedIn and Twitter.