

10 Lessons for ISVs to Consider when Containerizing Enterprise Software



The Shifting Landscape for Enterprise Software

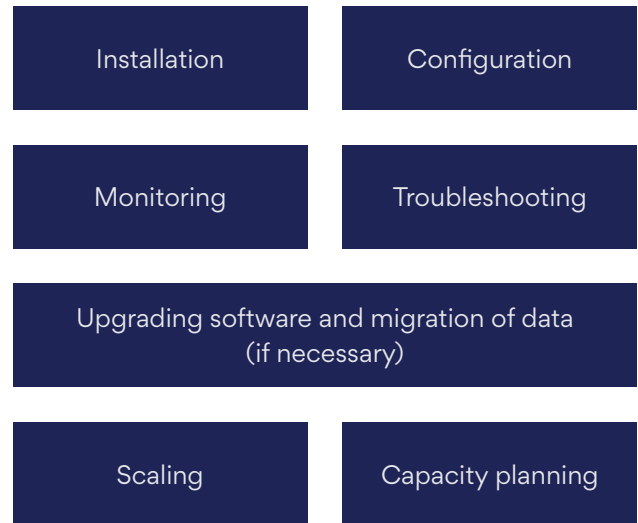
No matter whether your organization is focused on developing software offerings delivered through the cloud or on-premise, your total cost of ownership (TCO) is on the rise — representing a challenge both for the independent software vendors (ISV) and end-user customers. Complexity, in the form of upwardly spiraling requirements, concurrent users and large amounts of domain data — along with a host of other factors — is driving much of the trend.

From the operational expenses of running, maintaining, and administering software to infrastructure spending, those increased costs negatively impact existing customers (through rising license or subscription fees) and limit the ability to win new customers — ultimately eroding your profitability. You need solutions that can reduce your TCO growth and help you move to a more variable, controllable, on-demand model that maximizes your ability to drive new revenue streams.

As your customers clearly know, there is an inherent difficulty in making legacy software more efficient — a fact that plays out daily within the 80% of big deployments that run in the field in traditional IT sectors. Legacy software, developed in the on-premise data center era, typically have a large footprint in compute, memory, storage and networking, making it a suboptimal solution for modern needs and options.

In particular, the massive growth in networking requirements have compounded the issue. Today's collaborative teams, spread across the globe, are generating volumes of network requests that developers twenty years ago couldn't have imagined. This unforeseen increase is unnecessary slowdowns in low-latency networks, creating frustration for developer and user alike.

Successfully managing software instances is also a growing concern, due to the rapid increases in software complexity that are driving up costs in all aspects of software lifecycle support. Costs can rise in each of the following areas, including:



The advent of cloud solutions offered hope in mitigating those costs, but in many cases, they are neither the only answer nor even the best answer. Cloud costs have risen dramatically in recent years, bringing with them a host of additional challenges, including security concerns, configuration complexities, and migration difficulties.

To address the seven challenges listed above, automating this complexity is paramount. It needs significant improvement in those workflows and user experiences to hit that sweet spot.

Besides, today's customers want options, whether delivered via the cloud or on-premises. If your development approach commits you to one delivery model or the other, it will become increasingly difficult to win new customers who expect choice.



Solution search: SaaS/managed service, app modernization ...or both?

Software-as-a-service (SaaS) is a simple, straightforward way to provide efficient service with 24/7 availability, particularly in situations where significant technology upgrades and other investments would be required to reduce TCO.

There are downsides to this approach, however. While lift-and-shift to the cloud may have worked for simple applications, benefits drop off significantly for more complex applications — not to mention the high cost of third-party management in cases where SaaS operations and management are being outsourced.

App modernization can be a suitable solution that accommodates growth in the cloud platform and/or if there is a business need (e.g, security) on-premises. Beyond providing dedicated processing ability and isolated compute power, here the downside falls on the management aspect: There must be the desire and capability to manage the software.










Realistically, however, most ISVs today need to be able to deliver both: a viable SaaS or managed service offering that provides a competitive cloud entry point helps small to mid-sized customers, as well as app modernization for customers who also require an on-premises solution for security or other business reasons.

As the market evolves, containerization bridges that gap: a fully optimized, on-demand, and pay-as-you-go solution that offers flexibility along with durable improvements to TCO. The baseline concept here is that the software must be cloud-ready for SaaS offerings and managed services, while also working seamlessly in the on-premises world. Ultimately, the goal is to make complex things automatable so that administrators do not need to be knowledgeable in the minutiae of all the technology details.

For ISVs, the investment in cloud-readiness means building on technologies that are cross-platform capable: The ability to run your software on any cloud provider is critical, and being independent of infrastructure and runtime gives you and your customers more flexibility to be successful. It also accommodates the level of flexibility that companies are demanding for efficiency and productivity while giving the development organization full autonomy over where they are investing, what infrastructure they choose, and what platform the containers run on.

We are already seeing this shift to IaaS, PaaS, and container-as-a-service (CaaS) providers — with companies authorizing individual business units to provision on different cloud providers and form centralized operations teams to manage their projects on those cloud platforms.

10 Containerization Lessons Learned

Consulting Service	Strategy	Migration	Development	Management			
Advanced Technologies	AI	Analytics	Blockchain	Security	IoT	Quantum	
Cloud Native Software	Cloud Pak for Applications	Cloud Pak for Data	Cloud Pak for Integration	Cloud Pak for Automation	Cloud Pak for Multi-Cloud Management		
Foundation	Open Hybrid Multi-cloud Platform						
	 Red Hat	 RED HAT OPENSIFT	 Red Hat Enterprise Linux				
Infrastructure	IBM Public Cloud 	AWS 	Microsoft Azure 	Google Cloud 	Edge 	Private 	IBM Z IBM LinuxOne IBM Power Systems

Containerization has become a major trend in software development as an alternative or companion to virtualization. It involves encapsulating or packaging up software code and all its dependencies so that it can run uniformly and consistently on any infrastructure. The technology is quickly maturing, resulting in measurable benefits for developers and operations teams as well as overall software infrastructure.

The following list of lessons learned for ISVs has been formulated during Persistent’s five years of containerization development, implementations, and operations as an IBM Partner. While they apply specifically to our experience in containerizing products such as MDM CE and others, these best practices are also global enough to apply to any containerization strategy or hybrid multi-cloud platform environment.

Containerization Optimization



The basic idea is to continuously automate critical and complex workflows by containerizing, integrating, monitoring, metering and monetizing. The lessons are described below in these key categories.

Analyze and Prioritize Automation Workflows

- 1) **Focus holistically.** As noted earlier, the first priority is to identify the areas or processes that need to be automated, i.e. installation, configuration, monitoring, troubleshooting, upgrading software and migration of data (if necessary), scaling, and capacity planning. This is not a time to be siloed within your own organization or department. Input from your

customers and internal stakeholders is a critical piece of the puzzle, helping them prioritize the work required to automate these complex tasks and incrementally deliver them to market.

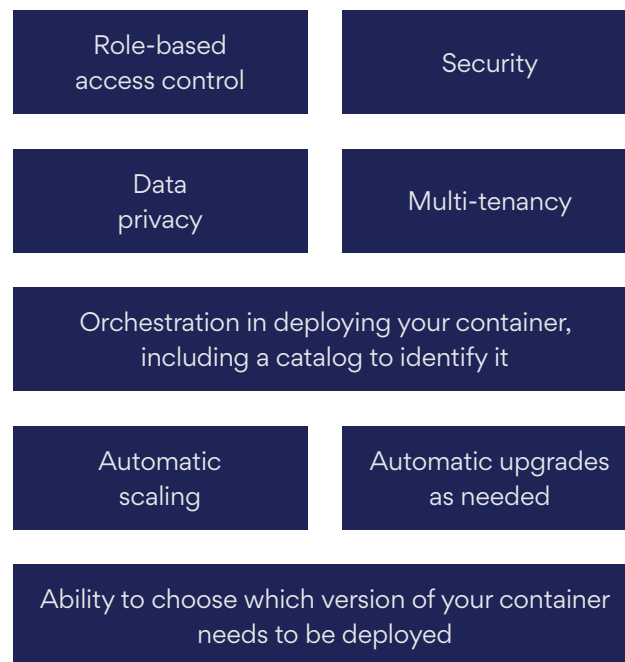
Package and Optimize

- 2\ **Minimize the container size.** The container base image must be simple and small enough that it can integrate well, taking advantage of process isolation and the ability to be moved easily. Take this opportunity to package only the required software dependencies in your container image to reduce the disk and memory footprint. This may require some tactical refactoring of legacy code to minimize the monolithic dependencies.
- 3\ **Align to the right standards.** Adherence to [Open Container Initiative](#) protocols will ensure that the standard-compliant containers you build can be plugged into standard-compliant platforms. Once you build a container image that follows the standards, it fits easily into container runtimes, regardless of the container platform you choose.
- 4\ **Keep data outside the container.** The best practice is to map data outside of the container, not inside. This practice makes challenging administrative use cases — such as backup, restore, and disaster recovery scenarios — much easier, because you can use standard data replication across data centers to achieve it.

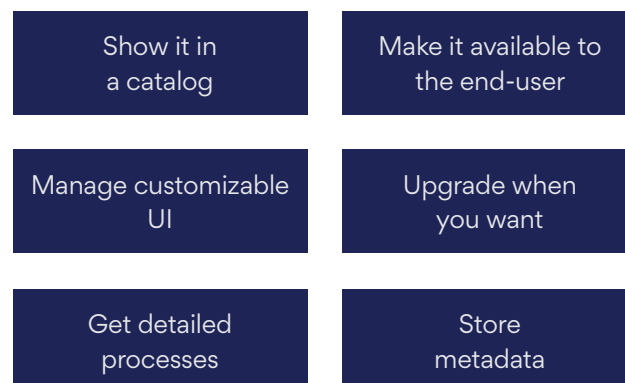
Integrate into the Platform

- 5\ **Explore simplification platforms.** Platforms with robust DevOps functionality, such as Red Hat OpenShift, can simplify the process and make it easier to manage different environments. Using your source control repository, they can build a code, build a Universal Base Image (UBI) and deploy it into a staging environment, run automated tests, and deploy into production. Other items that will come as part of a value-add offering, but are challenging to develop on your own from a free platform include:

These advanced features allow you to manage any number of containerized application instances without the need to code your own Kubernetes scripts, for example.



- 6\ **Think orchestration.** With a simple image and data partitioned outside, the next step is creating an orchestration layer on a platform where you can actually deploy. Many of the cloud providers offer a first-class service for container runtimes, although Red Hat OpenShift is becoming the default standard for Amazon, Microsoft, and IBM Cloud. Other options include Google, which offers its own runtime service for containers, or raw Kubernetes services offered, managed, and maintained by companies such as Microsoft and Amazon. The value-add of building an integration into a premium platform such as Red Hat OpenShift is that it allows you to define an operator that controls and participates in CRUD on your applications. If you choose a different platform (and you don't want to add to your development tasks and TCO), ensure that it includes the following value-add capabilities:



Monitor, Meter and Monetize

7\ **Don't forget to monitor performance.** While activating integrated application performance monitoring might sound like a daunting task, it can be a step-by-step process that becomes more sophisticated as your company progresses on its learning curve. Starting from the most basic level of performance monitoring, most services will send an alert with a note that simply tells you there is an application problem based on, for example, excessive CPU percentage. They may not understand why your product is not performing, however, and they won't fix it for you — so you're on your own for troubleshooting.

8\ **Instrument your code.** To address the next level up, understand the minimum characteristics you want to monitor and instrument the code accordingly, focusing on the items that have the greatest impact. Minimal instrumentation with key characteristics might include:

- \ Usage patterns (counts of projects, users, active users, concurrent users, amount of database growth, etc.), which allows you to prepare for the future, plan for capacity, and enforce best practices.
- \ Usage statistics on key workflows and license consumption, which can become candidates for the next round of automation and monetization.
- \ Performance metrics (compute, CPU memory, disk I/O transactional time, SQL server time, and average response times, etc.).

Red Hat OpenShift allows applications to participate in a centralized logging and monitoring system and contribute application-specific metrics and associated alerts. It also enables applications to use these custom metrics in defining auto-scaling criteria for your applications. Finally, keep in mind that containerizing something does not change the moving parts and the complexity — If you don't have meaningful data that you are bringing out to make decisions on, you still have an issue and trouble may lie ahead.

9\ **Maximize value with dynamic functionality.** One of the key aspects that needs to be addressed is the ability to dynamically control the compute parameters that support your software. For example, a built-in rapid allocation and metering system that controls price based on compute characteristics allows customers to pay for what they use, which makes it far more attractive to them. This method allows you to provision small, medium and large deployments affordably, with the ability to accommodate on-demand increases in size as well as capping them off. Note that this value-add is available in IBM® Red Hat® OpenShift®, which allows you to take the notion of a project and employs resource quotas to define compute constraints for a given project. This allows an ISV to have entry-level offerings with fixed computes at a lower cost, eventually getting customers to migrate to a pay-as-you-go model. It also gives more visibility and accuracy to central operations teams as they allocate costs for their internal business units for resource usage.

10\ **Align to new pricing models.** To enable efficient monetization of core capabilities, you want to move from traditional licensing models to token-based licensing, in which the base features of the software and automated administrative workflows can be based on tokens. This provides increased flexibility to customers who have highly variable patterns of software use within various projects and thereby enabling “the more you use, the more you pay.”

Benefits of Containerization

In the past few pages, we've outlined some of the trends and opportunities in containerization, as well as the challenges and best practices to consider when implementing a hybrid multi-cloud strategy. In closing, let's view containerization within the framework of where we started: the critical importance of addressing your growing TCO and complexity.

Here's a brief topline rundown of how containerization can benefit your customers —and thereby provide ISVs like you with a more compelling value proposition during the sales process:

Better time to value. Containerization represents the next generation in the hybrid cloud universe, offering improved resilience and repeatability of installs and upgrades for your end customers. In addition to the performance advantages, it reduces the skill, expertise, and effort needed to set up and maintain complex distributed environments. All of this translates to easier tilt-ups, faster time to market and better profitability for your customers — which means happier customers and improved sales and profitability for you.

Increase efficiency. By reducing redundancies and simplifying complex processes, containerizing enables your customers to lower their administrative costs, operations and IT staff requirements, and machine costs. Containerization also represents a more sophisticated and streamlined approach to multi-tenancy than virtualization alone.

Improve qualities of service. Containerization offers across-the-board improvements that your internal developers and your customers will appreciate — scalability, stability, self-healing, and dynamic capacity adjustment. By reducing complexity and potential problems, it can also serve to shorten admin outage windows.

Ideal for hybrid cloud deployments. Containerization eliminates the challenges of trying to adapt different software for different clouds or providers, whether on-premise, private clouds, or public clouds. For IBM Cloud users, there's an added benefit of self-service capabilities support.

One version, two models of delivery. Because your

SaaS offerings and on-premises offerings are built on the same architectural principles, there's no need to have or maintain two different versions of software — which means more profitability from a single piece of intellectual property. Build it once, re-use it in different contexts, and launch future updates without the redundancies or confusion that can occur from parallel development tracks.

Move from fixed licensing to token-based licensing. Legacy software has long been restricted to fixed or flex-based licensing — leaving money on the table and not always adequately charging customers for their usage. Containerizing enables you to charge for every value-add that's built in, even at finest level of automation.

Not only does that increase your profitability, it's a more transparent payment system for customers, because they know what they are spending. Bottom line, for every incremental value-add and new release that's built on the platform, token-based licensing increases the potential revenue.

Improve your competitive position vs. pureplay and legacy-free ISVs. Containerization opens your doors to being more flexible in how you are competing. By leveraging the best aspects of SaaS and on-premises delivery methods, you can be more agile and responsive — providing your customers exactly what they need, exactly when they need it — at a lower TOC than ever before.



About the Author

Vishy Ramaswamy is a distinguished engineer in the ELM development group of IBM Alliance Division in Persistent. In this capacity he is responsible for defining and managing the overall architecture of the IBM® Engineering Requirements DOORS® Next Generation product and the lead architect for project Slipstream, which is the modernization and containerization initiative for all of the IBM IoT ELM Product Suite, the encompassing DevOps strategies and associated value-add administrative workflows. Vishy has been working on containerization-related initiatives for the last five years.

Vishy's career in software development spans 24 years, during which time he has worked in a technical leadership role on products in the application lifecycle management space and software services related to the telecom, wireless, health care, federal government, and defense industries.

About Persistent

Persistent Systems (BSE & NSE: PERSISTENT) builds software that drives our customers' business; enterprises and software product companies with software at the core of their digital transformation.

www.persistent.com

India

Persistent Systems Limited
Bhageerath, 402,
Senapati Bapat Road
Pune 411016.
Tel: +91 (20) 6703 0000
Fax: +91 (20) 6703 0008

USA

Persistent Systems, Inc.
2055 Laurelwood Road, Suite 210
Santa Clara, CA 95054
Tel: +1 (408) 216 7010
Fax: +1 (408) 451 9177
Email: info@persistent.com



Persistent