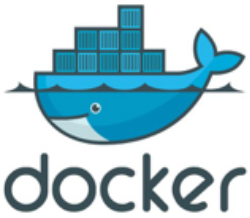
Enterprise Software Delivery in the Age of Docker and Other Containers

Executive Summary



Most of today's forward-thinking companies are investigating container technology as a potentially better way to deliver software. Containers let developers bundle all of the necessary components of an application into one package that can easily be shared and distributed, which can greatly simplify deployment to test, user acceptance and production environments.

However, containers by themselves solve only a fraction of the challenges enterprises face as they try to deliver high quality software, quickly and securely, with adherence to compliance standards and industry regulations. The usual release challenges do not disappear just by implementing containers. And in many cases, containers create additional complexity and dependencies that must be managed. In addition, it will be a long time (if ever) before enterprises work exclusively with containers. So for many years to come, companies still need to manage diverse infrastructures and hybrid environments. Containers are a great technology for software delivery, but if you want to use them at enterprise scale – beyond individual green-field experiments – there are critical management considerations that you can't overlook, or you will create a painful and costly minefield.

So how do companies take advantage of container technology and at the same time consider their broader needs? To be successful with containers in the enterprise, they need to be able to scale with a consistent deployment process across hybrid environments. They also need to carefully manage and orchestrate the entire release pipeline. And they need visibility into the overall release process, including status of all components, dependencies between parts, and which versions and configurations are present in all environments. And although containers give developers the power to package software for deployment, enterprises still need to enforce control over the delivery process and log all activities to ensure compliance. As companies move to container technologies, these enterprise needs must be addressed.

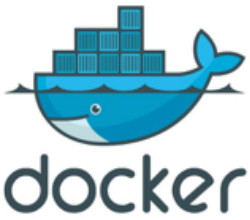
The Benefits of Docker and other Containers

The usual release challenges do not disappear just by implementing containers.

Docker, Mesos, Kubernetes and other container management tools allow you to package an application into a standardized, self-contained unit for software deployment. Containers include everything the application or a service needs to run: code, runtime, system tools, and system libraries in a form that is easily installed on a server. The contents of the container run in a protected execution space, isolating the application or service and making it easier to manage. Because container formats are portable, any execution environment that supports a container standard like Docker can run the container.

Containers represent a "new paradigm" in which applications and services can be delivered as sets of versioned containers by delivery teams. Companies are intrigued because containers…

- Give developers more control by simplifying application delivery
- Are more initially responsive because they start up faster (with no Virtual Machine and host OS to boot up, they can start/stop in seconds)
- Have a smaller runtime and storage footprint than Virtual Machines, which reduces storage and network bandwidth needs
- Are an enabling technology for microservices architectures
- Provide horizontally scalable components, potentially making it easier to load-balance the application or service
- Let you better manage to a set of operational resource constraints by setting resource utilization limits at the container level (good for cloud architectures)



Where Containers Need A Boost

Containers are ideal for pushing out small-scale applications that aren't constrained by enterprise requirements. However, containers are currently unproven for highly reliable, highly scalable production applications. There are many unresolved issues, and with the overall market shifting rapidly, it is very difficult to figure out which tools and approaches are likely to survive in the long run. By themselves, the available tools and frameworks generally don't offer everything an enterprise needs in order to release high quality software quickly and at scale.

As enterprises progress beyond the initial investigation of containers, they discover there is more to orchestrating multiple, interdependent containerized applications than they expect – whether they're using Docker directly or one of the many container orchestration frameworks. Most discover very quickly that containers are not a complete solution, and they realize that they have to accommodate a mixture of containers and traditional application environments.

One of the biggest benefits of containers is that some deployment technicalities become easier: developers can "ship" the application code together with the operating environment. However, the actual release process doesn't go away and the typical challenges faced throughout an enterprise-scale software delivery process stay the same. Containers don't offer a few key capabilities that are critical to meet enterprise demands.

Containers are a great technology for software delivery, but if you want to use them at enterprise scale – beyond individual green-field experiments – there are critical management considerations that you can't overlook, or you will create a painful and costly minefield.

1. Standard and Repeatable Processes for Hybrid Environments

Today, containers exist in broader scenarios that also include Virtual Machines and traditional environments (a.k.a. hybrid environments). Container adoption is so nascent that it will be a very long time, if ever, before an organization only needs container management tools. They will continue to need to support components like databases and systems that are not containerized and that live in a variety of different environments.

Organizations still need a standardized, repeatable way to deploy software across all environments. Release processes must be followed and rules must be enforced, whatever the underlying technologies. And enterprise teams need a deployment model that automatically adjusts for any environment, following consistent processes across all targets.

2. Dependency and Complex Process Management

Containers can show quick value for simple deployments, but as you introduce multiple interdependent services and scale, you will find that the complexity can quickly spiral out of control and must be managed.

For example, Docker and containers are very often used for microservices architectures. As applications are broken into smaller chunks, it's critical to be able to manage the relationships between its services and components. Microservices architectures have many more moving parts and often involve more releases, so there is a much stronger need for release and deployment platforms that manage dependencies and give a holistic view of what's happening in your process: think tens or even hundreds of services coming together to deliver the full business application that will be running in production.

In addition, today's DevOps processes not only have dependencies between components in a release, they have multiple interdependent pipelines that require advanced release processes, such as release trains and master/sub-releases.

3. Compliance, Security, Reporting, Audit Trails, and Process Control

Science experiments and pilot programs may be able to ignore enterprise considerations, but production environments can't. Core enterprise capabilities like compliance and control measures, enterprise security enforcement, process control, audit trails, and the ability to generate reports are missing entirely from most of today's containers and frameworks. You need a way to make sure that processes are followed and logged, and that compliance, governance and security are enforced for all applications and components…. as well as for new deployment targets as they come online.

Further complicating matters, containers wrap up everything into one package. But when you deliver "the whole machine" each time, you have less knowledge of what's really "in" there. It's difficult to control what the developer did, and this lack of control opens companies to security and governance problems: what is delivered may or may not be compliant with the applicable rules and regulations. And if a company needs to do a SOX or PCI audit, they'll have to audit the entire container.

4. Real-time Visibility into Processes and Components

Containers simplify deployment packages, but they don't simplify release processes… and in fact they often add complexity as you introduce more moving parts and release more quickly. To bring order to the chaos, people from across the organization, from Development to Operations to Management to Security – both technical and non-technical – need to be able to easily tell what's going on, where and when, across all systems. Teams need an integrated dashboard that shows real-time status of all application components, release processes and deployment artifacts. And if they are committed to continuous improvement, they need to be able to capture metrics across the whole pipeline.

In addition, containers often have different OS configuration and middleware versions, as well as different versions of the same app. It's difficult to know what's running a particular container, unless you use very technical interfaces (picture hundreds of lines of log files). As versions of components multiply and environments proliferate, it's especially important to know which versions are deployed where, the current status of all components in an environment, and what the environment configuration looks like. Think about how you will answer a simple question such as, "Which version am I testing?! And what does the environment look like?"

> **As app code complexity declines, managing service dependencies looms.** As applications move from being tightly-coupled monoliths to loosely coupled networks of services, organizations will need better means for managing them.
>
> Applications will be composed of potentially hundreds or thousands of services, sourced from different vendors and running in many different environments, with dependencies managed at runtime. Versions matter, as a particular version of an application is dependent on particular versions of the services it uses.
>
> Managing this complexity is beyond human capability, and new tools will be needed to help.
>
> *Forrester Research, Feb 2015, Application Design To Application Composition: How APIs, Micro-Services, And Containers Are Changing The Way Organizations Develop And Deliver Software*

Harnessing the Power of Containers for Enterprise Application Delivery

It's nearly impossible to scale to handle complex container scenarios without collaborative, cross-departmental tools to help orchestrate releases and automate deployments.

Forrester Research analysts Kurt Bittner and Michael Facemire write in their February 2015 report, *Application Design To Application Composition: How APIs,*

Micro-Services, And Containers Are Changing The Way Organizations Develop And Deliver Software: "Managing this complexity is beyond human capability, and new tools will be needed to help."

Release Orchestration and Deployment Automation tools can play a vital role in bridging the gap between the promise of containers and the realities of complex enterprise application delivery. These tools provide key functions that IT teams need to complement container use in diverse enterprise environments:

- **Standardization, automation and control** of complex software release pipelines and deployment processes
- **Dependency management** between applications and between release processes
- **Complete visibility** into the software delivery process and deployment status across all environments
- **Compliance, security, reporting, governance, and audit trails enforced** throughout your release process
- **Hybrid deployments** managed across a mixture of containers, VMs, and traditional environments
- **Release and deployment information that is easily accessible** across all teams, both technical and non-technical

Conclusion: How to Address the Enterprise Challenges Presented by Container Technologies

Docker and container frameworks are up-and-coming technologies that can greatly improve software delivery for large companies and regulated enterprises, provided they are well managed and implemented with enterprise requirements in mind:

1. Standard and repeatable processes that work for hybrid environments
2. A good way to manage and orchestrate complex processes and dependencies
3. Infrastructure to address Compliance, Control, Security, Reporting, and Audit requirements
4. Real-time visibility into all aspects of release processes and components

As companies continue to accelerate development and release software at enterprise scale, they can address these requirements with Release Orchestration and Deployment Automation tools, which manage and control the many moving parts in their pipelines and environments.

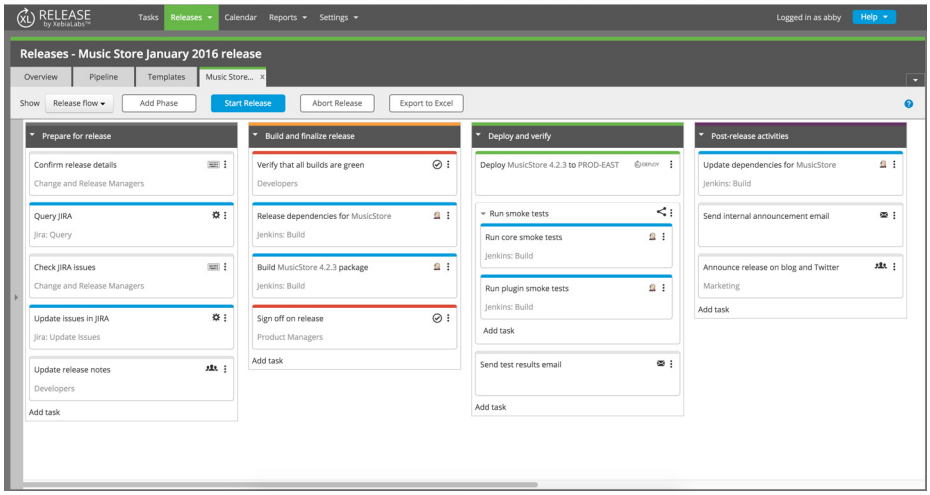
XebiaLabs and Containers

XebiaLabs tools complement container technology and help companies achieve Continuous Delivery at Enterprise scale.

XL RELEASE

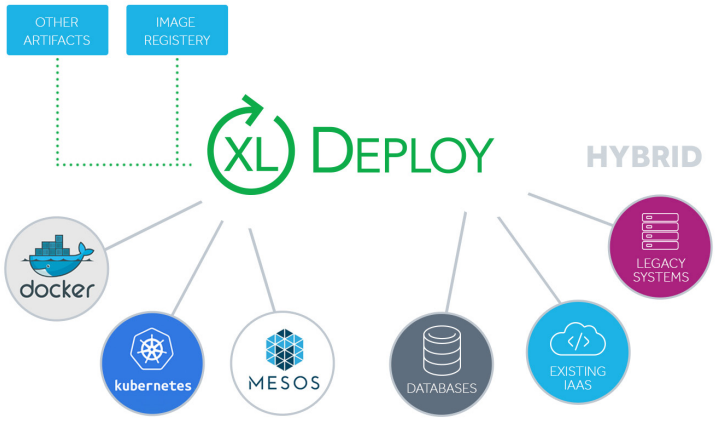
Where in the pipeline are new versions and features currently? How do they depend on each other? Where are the bottlenecks in my process?

XL Release lets you automate, orchestrate and get visibility into your release pipelines – at enterprise scale. It allows you to easily define and run delivery pipelines for multi-container, microservice and hybrid scenarios, with templates for decoupled release trains, coordinated master and sub-releases, and other advanced release patterns. Rich dashboards, release metrics and audit trails give you confidence that processes are followed and that your team is kept informed about the status at each step. As you scale, you need XL Release to manage your complex release pipelines and dependencies between components.

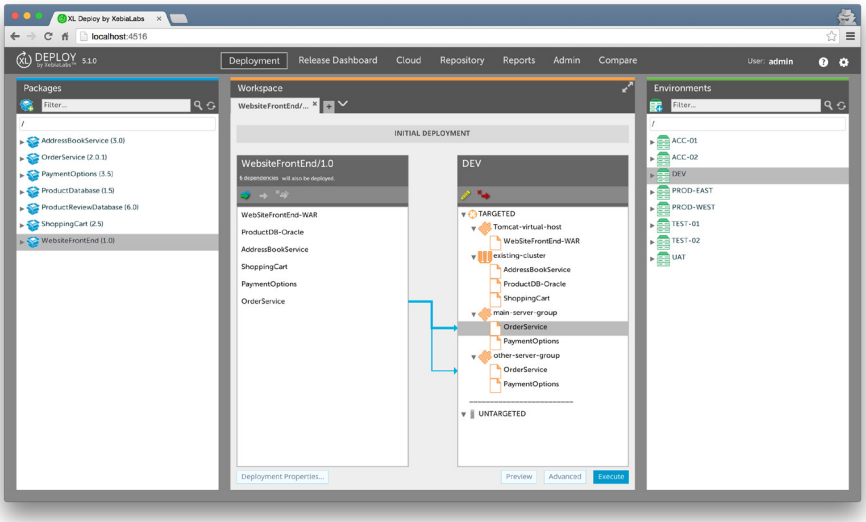


XL DEPLOY

Where are all versions of my current services running? Are they all compatible with each other? How do I know my deployment process is consistent across all environments?



With its agentless architecture and model-based approach, **XL Deploy** is the most advanced deployment automation tool available today, automating and standardizing complex deployments to any target environment. It automatically handles complex multi-container and hybrid deployment scenarios, integration with your Continuous Delivery pipeline, environment-specific configuration, rollbacks, and more. XL Deploy lets you handle multi-container and deployments to any target with ease. And once your deployment tasks are modeled in XL Deploy, you can be sure they will work repeatably and will automatically adapt to new targets.



About XebiaLabs

XebiaLabs develops enterprise-scale Continuous Delivery and DevOps software, providing companies with the visibility, automation and control to deliver software faster and with less risk. Global market leaders rely on XebiaLabs to meet the increasing demand for accelerated and more reliable software releases.

For more information, please visit www.xebialabs.com.

Continuous Delivery at Enterprise Scale

Orchestrate, automate and get visibility into release pipelines

Automate and standardize complex application deployments

Analyze test results across multiple test tools

→ Learn more and try for free at www.xebialabs.com