CONTAINERIZATION

ENABLING AGILITY ACROSS THE SOFTWARE VALUE CHAIN

Table of Contents

EXECUTIVE SUMMARY

From AI-driven analytics to blockchain to serverless computing, the rate of technology change today is jaw-dropping. At the same time, the digitization of IT processes and commerce is accelerating the speed of business to the point where the “need for speed” has become a tagline for digital competition.

The strain of keeping pace is leaving a trail of major brands that failed to innovate like AOL, Atari and Enron Online. However, the struggle itself has also become a key driver of innovation and experimentation in infrastructure and application development strategies.

While “hot” initiatives like digital transformation and IT modernization are top-of-mind with corporate IT leaders, another movement is gaining momentum based on a decades-old process-isolation practice that involves changing the root directory for a running process and its children. As underwhelming as this sounds, a lot can happen in 40-plus years.

Fast forward to the age of DevOps, hybrid cloud and hyperconvergence and we are reminded that in technology, as in science, old concepts can become newly relevant in different times and environments. It is the premise of this paper that the somewhat dry concept of isolation or abstraction has evolved into an exciting new incarnation called “software containerization” at exactly the right time.

In the following pages, we will explain how containerization accelerates application development, delivery and modification and enables “extreme portability” across today’s multi-cloud and hybrid IT infrastructures. We will also explore how the unique services and cloud-native development practices of cloud providers and independent software vendors are helping to promote container adoption.

Like every important new strategy, the use case, success factors and dependencies of containerization will be unique to each business and development environment. However, exploration of some common factors and benefits may prove useful, in combination with the customer use cases and links to in-depth articles and expert analysis provided herein.

By 2023, more than 70% of global organizations will be running more than two containerized applications in production, up from less than 20% in 2019.[1]

SECTION I
MARKET OVERVIEW

To thrive and grow in today's markets, every enterprise – from online retailers to financial services providers to independent software vendors (ISVs) – must perform at digital speeds, with the efficiency and ease that modern customers require. However, the "secret sauce" to consistent market success is business agility.

More than a fleeting buzzword, the concept has its genesis in software development as part of the agile framework – a means of addressing the problems caused by the changing requirements and increasing complexity of modern IT projects, infrastructures and market dynamics.[2]

So, is business agility simply the need to respond and adapt to change? Isn't that what employees try to do anyway?

Of course, true agility can't happen unless teams perform; however, the productivity engine of every modern enterprise is software. The faster, smarter and more efficient the apps, the more agile the business can become. Likewise, the more efficiently ISVs can package and deliver software to their enterprise customers, the more productive and profitable they become.

Thus, it comes as no surprise that the concept of "agility" ends up where it began - with software development.

Containerization comes of age

In the midst of often frenzied IT initiatives, a new form of containerization is taking center stage as a potential building block of infrastructure modernization. The benefits are compelling; however, there are myriad considerations on the path to successful implementation.

For many companies, containerization is a tectonic shift in how they design, build, package and ship software, and there will be obstacles to overcome. For ISVs, however, it is a necessary strategy to accelerate time to market – and time to value – for their customers, most of whom need to run applications on a variety of platforms. As enterprises begin to catch up (and they will), the benefits of increased business agility will accrue across the entire software lifecycle.

"2020 will mark the year that container-centric initiatives become the go-to-approach and play out on a larger scale, across enterprises and industries…providing a clear path to the cloud, while reducing cost and risk."

Scott Johnston, CEO Docker[3]

SECTION II
UNPACKING CONTAINERS

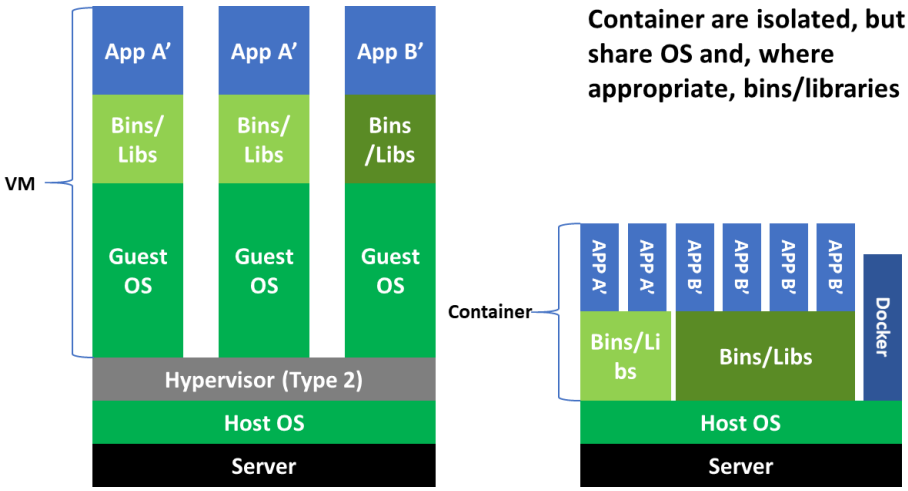
Anatomy of a container

Simply put, a container packages up the application code and dependencies, libraries, binaries and configuration files needed to run it. However, instead of deploying a full copy of the stored container image, the copy-on-write mechanism enables sharing of read-only versions of the host OS kernel, binaries and libraries, so multiple containers can share the same OS – and VM or bare metal server.

Able to boot up in seconds, correctly configured containers can be deployed in a wide range of environments, making them ideal for today's hybrid and multi-cloud infrastructures. Additionally, developers can use the same tools and workflows, regardless of target operating system, making it easier to manage and move applications.[4]

Containers vs. virtual machines

For decades, software architects and developers have been trying to break monolithic applications into smaller components to enable true agility, enhanced modification and portability. Virtualization moved the goalpost by consolidating OS resources such as CPU, system memory and disk storage in virtual machines. However, because VMs carry a copy of an operating system, they consume considerable memory and disk space, making them bulky and slow to boot.[5]



A VM can measure several dozen gigabytes, while a container usually measures only a few megabytes[6]

Containerization also enables scalability from 10 to 100 times that of VM environments.[5] And while virtual machines can move between systems running the same hypervisor, containers can move wherever there's a copy of the host OS – no hypervisor needed.

Explore the Xoriant blog

Explore the Xoriant blog How to Select a Modern Container Orchestration Platform for Your Cloud Apps

Key containerization benefits

As the latest form of virtualization, containers provide the development-cycle speed, scalability, agility and portability that ISVs and corporations need, enabling:

Enhanced process and engineering productivity

IT leaders are able to commit more resources to innovation, enhancements and core development activities.

Extreme portability

Containers allow easy movement among diverse environments and cloud platforms (write once, run anywhere), while avoiding vendor lock-in.

Easier management

The install, upgrade and rollback processes are built into the container platform for greater speed and efficiency.

Agility

Containerization facilitates integration and modernization of DevOps environments, while providing a path to microservices and agile service delivery.

Faster delivery

Engineering teams can deliver enhancements and execute changes and upgrades quickly for higher productivity and profits.

SECTION III
THE CONTAINER ORCHESTRATION IMPERATIVE

To operate at scale, enterprise infrastructures need to automate the deployment, management, scaling, networking and availability of containers. However, when applications are containerized in a cloud-agnostic (versus a cloud native) environment, they lack the necessary plumbing support and services to enable the horizontal scalability, high availability and resilience required of modern architectures. Equally important, developers lack the container orchestration tools they need to define the policies for host placement, security, availability and performance. [7]

Orchestration tools enable the agility benefits of containerization by allowing developers to run containerized apps when and where they are most needed and/or can function most efficiently, and:

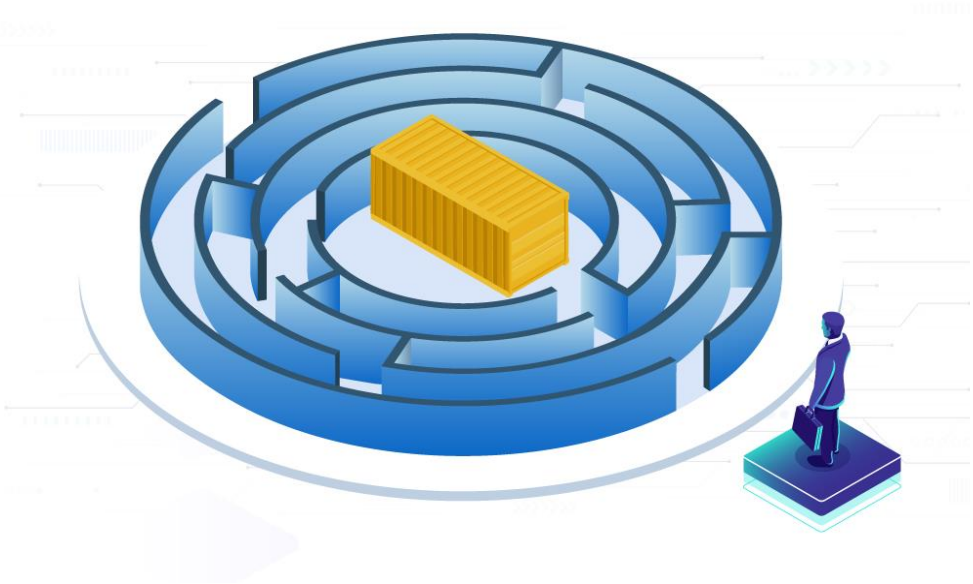
Provision and deploy containers quickly and easily

Externalize configuration of containerized apps

Continuously monitor container health (healing)

Launch and/or scale a fixed number of container instances

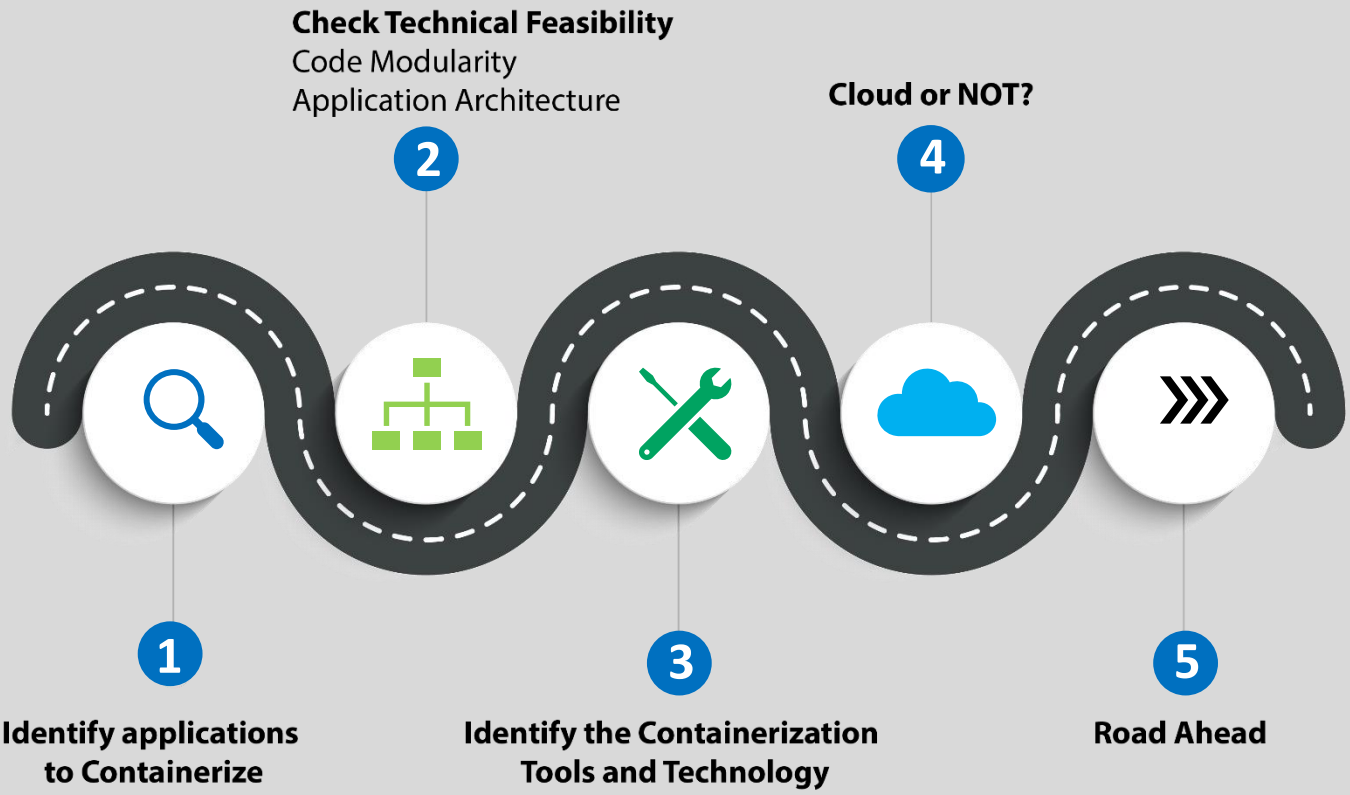
Expose container-hosted services to the outside world



SECTION IV

CONTAINERIZATION STRATEGIES, TACTICS AND TRENDS

While the benefits of containerization may be compelling, strategy and implementation are neither simple nor static. They must be customized for each unique IT environment and evolve to keep pace with the technological changes of this era and the next. Accordingly, developing the right strategy will hinge upon consideration of these factors:



1. What to containerize and why?

Stateless applications top the list because the configuration information is stored locally (i.e., web applications that use a backend for persistence and have tiers that only perform processing). If stateless tiers can be peeled off, they provide enormous flexibility. Likewise, apps where one can segregate the components with system boundaries are good candidates.

Scalability and healing ability are also important factors in the selection process; for example, apps that need to scale differently or auto-scale/heal to perform optimally. Another consideration could be high-maintenance, system-critical legacy applications that require increased flexibility and extensibility to meet business needs; or apps that must provide a superior user experience and are therefore subject to frequent updates and releases.

Finally, examples of 'low-hanging fruit" may be applications that are already part of a CI/CD process, as well as greenfield apps that can be designed from the ground up to leverage cloud-native attributes.

2. Technical feasibility considerations

After identifying the candidates for containerization, the next step is to explore the technical feasibility, which involves selecting the optimal tools and technologies to use in a specific case. For each of the various application tiers, determine if the necessary docker images are available or can be built from scratch (e.g., UI tier, application server tier and backend database tier). It's also important to look at CI/CD tools and make sure they have the necessary support for integrating containerized applications. Finally, evaluate and identify the best monitoring tools for your applications once they are containerized.

3. Containerization tools and technologies

For those well-versed in container technologies, an exploration of options like the following can be found later in this paper. For example:

- Is Docker-compose the best, most future-proof option?
- Is a transition from YAML files to Helm-based deployments on Kubernetes the right move?

The sheer volume of considerations can be overwhelming, especially for teams just beginning to strategize. While brands like Kubernetes and Docker have become mainstream, they are constantly changing and evolving.

For all teams, a word of caution: be mindful that what may work for a cloud-native startup may spell trouble for companies if legacy systems, age of infrastructure and governance standards are not taken into consideration.

According to a recent IDC survey, 85% of container adopters are using containers for production apps, and 83% of deployers use containers on multiple public clouds (3.7 clouds on average), with 55% of containers running on-premises and 45% running in the public cloud.[8]

4. Cloud factors – public, private or hybrid?

As more companies migrate workloads to the cloud, their teams evaluate the pros and cons of public versus private cloud for each use case; in particular, how it may support (or impede) their business and rapid application development goals for increased agility, resiliency and reduced costs.

Cloud native vs. cloud agnostic

What are the choices? Commit to using one cloud provider and leverage the resilience, availability, services and resources of going cloud-native, but risk vendor lock-in? Or employ a cloud-agnostic approach using open source tools and standards within the on-premises infrastructure? While the latter can make migrating from one cloud to another relatively easy, it can be difficult and costly to build resilient, highly available architectures without the unique services and platforms that cloud providers offer.

Deployment and orchestration platforms

Depending upon whether one chooses the private or public cloud option, enterprises must identify the optimal target container-orchestration platform. Teams using private cloud have been leveraging Docker Enterprise or Kubernetes cluster; but many are now moving towards PaaS platforms such as RedHat OpenShift or Pivotal Application Services due to the ease of administration and operational benefits

Please note that if one plans to create a Docker Enterprise or K8s environment, it is essential to assess the hardware resource requirements for compute, networking and storage up front.

Multi-cloud deployment options

Enterprises that need the flexibility of using both private and public clouds are leveraging containerized cloud-native designs and services that allow them to build product offerings able to be deployed to any target containerization orchestration platform. For example, on private clouds, they may opt for the RedHat OpenShift container platform, Rancher or Platform9-based Kubernetes; whereas on public clouds, it could be AWS-ECS, AWS-EKS or Azure-AKS.

Infrastructure	Container Orchestration / Deployment Platform
Public Cloud	AWS – Elastic Container Service (ECS)
	AWS – Elastic Kubernetes Service (EKS)
	Azure Container Instance (ACI)
	Azure Kubernetes Service (AKS)
	Google Kubernetes Engine (GKE)
Private Cloud	Red Hat OpenShift Container Platform
	Docker - enabled Pivotal Cloud Foundry
	Kubernetes Cluster
	Docker Swarm Cluster

As IT teams continue to work on their containerization strategies and implementations, it can be useful to learn from others' experimentation, successes and failures. To that end, the next section provides a summary of current containerization trends, followed by use cases employing a variety of approaches.

Explore the Xoriant blog

How to Select a Modern Container Orchestration Platform for your Cloud Apps.

5. The road ahead: tactics and trends

The common denominator for the sea change in application development today is that enterprises with decades-long histories of building monolithic legacy applications are moving towards containerized microservices-based architectures. The following trends and practices may help teams avoid some speed bumps along the way.

Increasing support for containerization

As discussed earlier, containerization is best fit for stateless applications. However, virtually all database (SQL and NoSQL) vendors are building Docker image support to provide containerized database solutions. Even in the big data space, ecosystem components such as Hadoop/HDFS, HBase and Kafka are getting containerized. Enterprises and Open Source communities are also building support for Kubernetes Operators to completely automate deployment and monitor related tasks with an easy-to-use declarative mechanism.

Deployment tools – current and future

Docker tools such as docker-cli and docker-compose are widely used today. For Kubernetes, YAML-based deployments are commonplace, but there is change now towards Helm Charts-based deployments. As a best practice, these tools are increasingly integrated with CI/CD tools such as Jenkins (free/open source), TeamCity (free scaled version) or Bamboo (most expensive).

Why CI/CD integration?

Integration with the CI/CD pipeline facilitates deployment of product artifacts to a wide range of end infrastructure services, which can be private- or public-cloud based. This agility allows product owners to build software products and add features faster with improved quality – if the following steps are taken:

- Build CD pipeline along with CI pipeline (CI/CD)
- Integrate deployment within CI tools like Jenkins, TeamCity with OpenShift, Pivotal Cloud Foundry
- Leverage cloud infrastructure services on AWS, Azure

SECTION V
CONTAINERIZATION USE CASES

About Xoriant

Xoriant Corporation is a product engineering, software development and technology services company dedicated to helping startups to global corporations compete more effectively. We offer a flexible blend of onsite, offsite and offshore services from our eight global delivery centers, comprising over 2500 software professionals.

The following use cases are a sampling of Xoriant consulting and implementation engagements. For more detailed information on any case, contact the Xoriant expert named at the end of this document.

USE CASE A.

Containerization creates high performing, auto-scaling architecture for financial services software provider

This leading financial services software company provides an online ads-impression reconciliation platform that automates the entire reconciliation process – from contract to payment – for the digital advertising industry. With offices in New York and San Francisco, it processes hundreds of billions of ad-campaign events in 23 countries.

The technology challenges

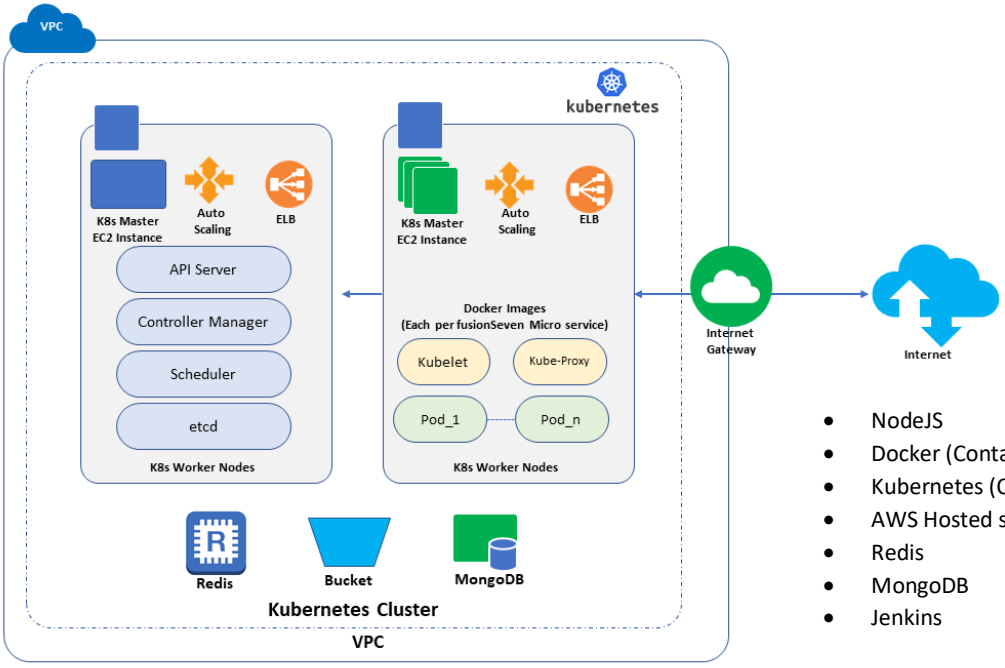
The current application for dealing with data reconciliation of online advertisements was processing around 1.2 million records per day, per client and showing signs of performance degradation – an unacceptable situation when revenues depend on transaction speed. The client's IT team concluded that the legacy application could not keep pace with the constantly growing number of records, and something had to be done.

Central to the problem were earlier decisions made while designing the microservices-based architecture using NodeJS microservices architecture and EC2 VM and SQS queue technologies, which lacked autoscaling capabilities. This limitation impeded invoice generation for reconciled data, negatively impacting the client's revenue.

The Xoriant solution

The Xoriant team set up a Kubernetes cluster with master and work nodes, created the Docker images and deployed the microservices inside an AWS-EKS cluster. The microservices were exposed using REST API through elastic load balancer (ELB) and the APIs were integrated.

New technology stack



- NodeJS
- Docker (Containerization)
- Kubernetes (Orchestration)
- AWS Hosted services: SQS, S3, EC2 VMs, CloudWatch
- Redis
- MongoDB
- Jenkins

Solution benefits

Application performance improvements as a result of the Xoriant engagement included reductions of 88% in reconciliation process times and 80% in single REST API call response times. Reductions were also significant in overall deployment costs and included a 77% savings in AWS EC2 instances alone.

By replacing manual analysis with automated reporting, client stakeholders continue to improve productivity, with significant time and cost benefits.

USE CASE B.

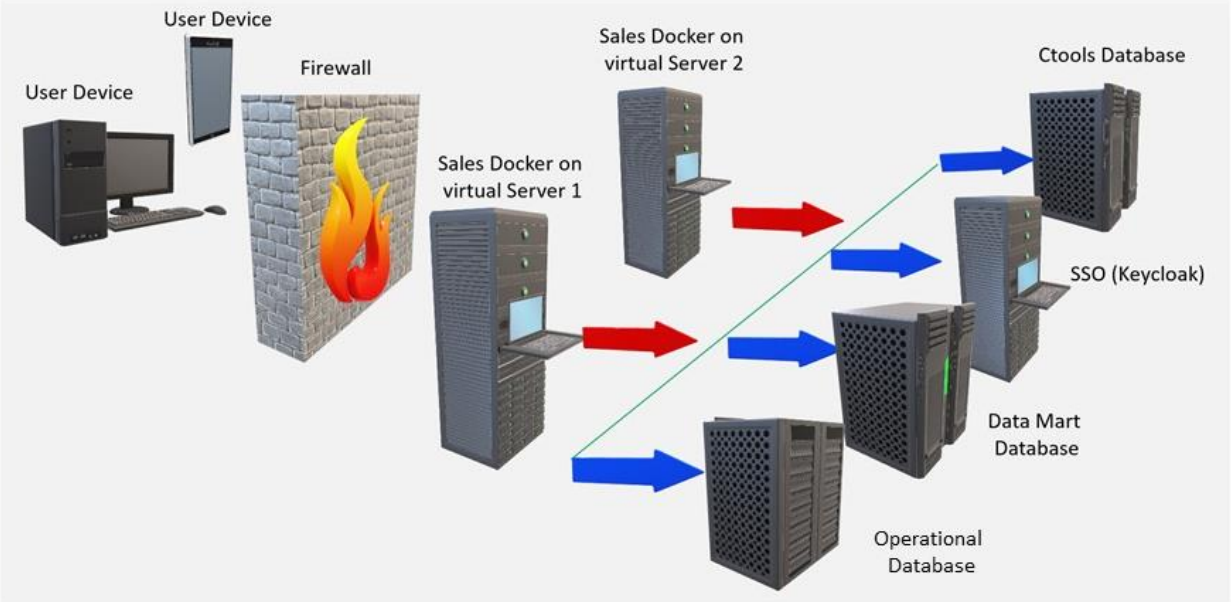
Containerization enables rapid deployment, configuration and future cloud migration for a cloud services division of a Fortune 100 tech company

The customer is a global leader in mobile interconnection and mobile consumer engagement services, and a cloud service group of a Fortune 100 company with 25 billion in revenues. It provides mobile operators with global-messaging interconnect, data roaming and IPX-based services that enable enterprises to bridge the “last mile” in order to engage their consumers. The company currently helps businesses process more than 1 billion messages per day, reaching 1,100 operators and 6.1 billion subscribers across 220 countries.

The technology challenges

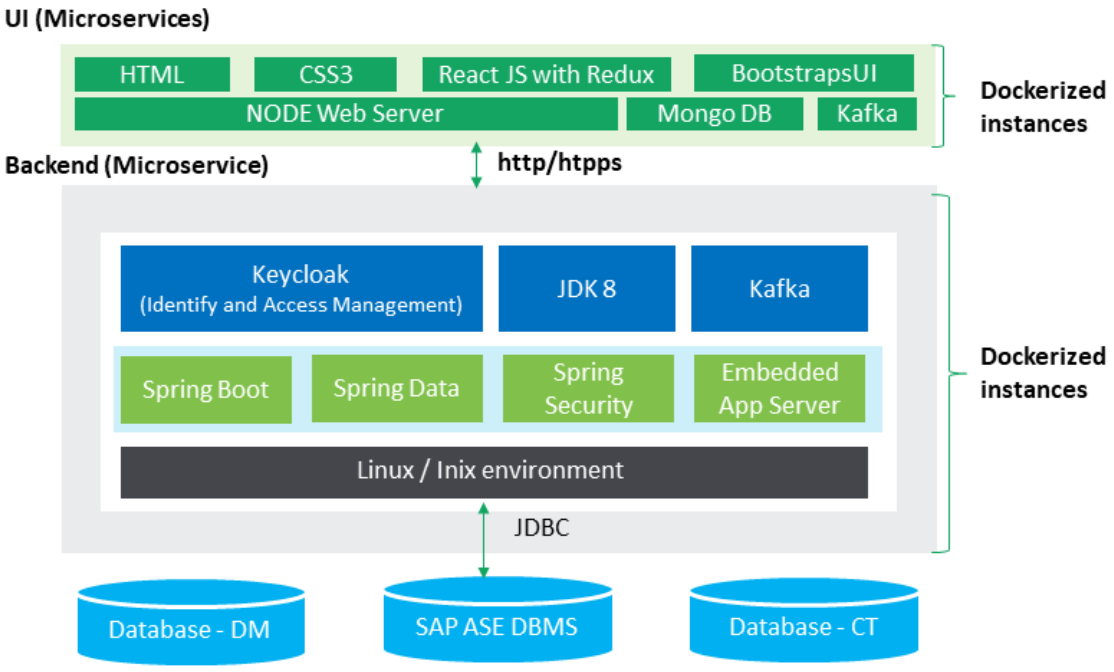
A major challenge undertaken by the client’s chief architect was the re-design of the main application architecture to provide a Responsive Front End for globally-dispersed sales and operations to access data from four systems of record from any location or device. These mission-critical databases had to remain monolithic due to security concerns and the complex layering of new and old information.

The task for Xoriant was to build containers and microservices to package the main application components for eventual hosting on one of the major cloud platforms. Containerization was also required to enable quick and easy deployment and configuration of the constantly changing functionality requirements of the main application.



The Xoriant solution

The team deployed the new run-time environment using Docker EE and approximately 15 containers. Containerization for the microservices was comprised of approximately 14 app-specific microservices (excluding databases).



Development entailed a manual build process using Bamboo (Docker image creation with push to target server).

USE CASE C.

Modernization of rigid, legacy infrastructure achieved using microservices and containerization of the client's payment hub publish, subscribe and transform services

The client's payment hub application is built on legacy technologies that accept transactional messages from Kafka/MQ/File, execute transformation and transmit to multi-channel upstream/downstream applications.

The technology challenges

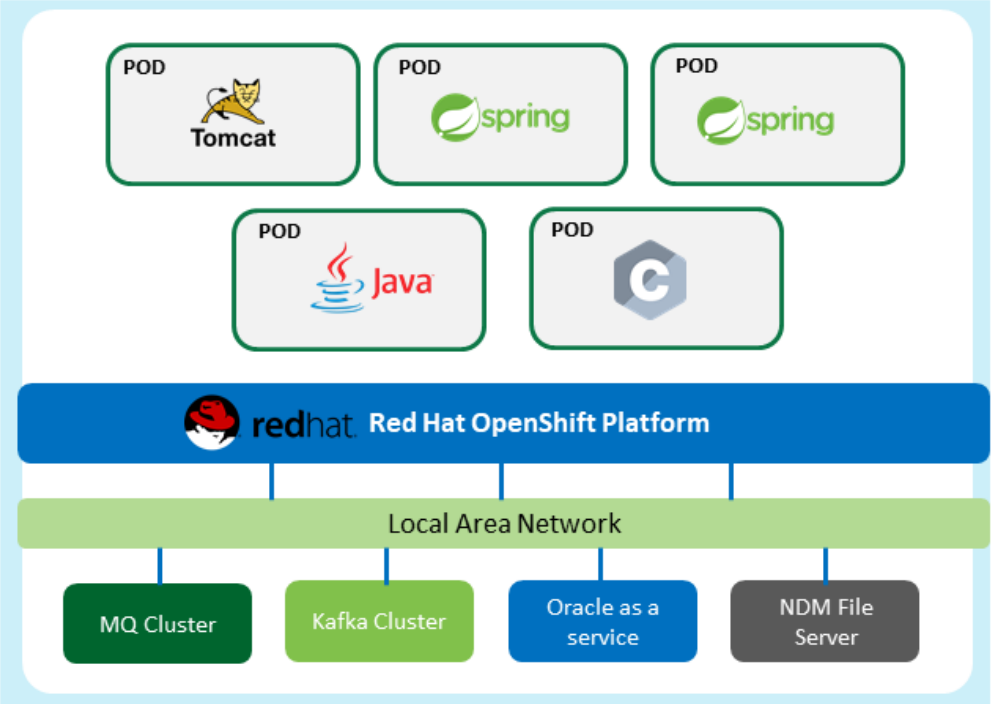
The legacy technologies in place lacked the key functionality of high availability modern infrastructures, including mission-critical services such as automated data reconciliation, message transformation, and event triggering and auditing. Also missing was SSL for Kafka and MQ message processing, and a UI to view and reconcile Kafka messages.

Xoriant Solution

The focused-scope project leveraged microservices and containerization to modernize the payment hub's publish, subscribe and transform services, accessible via a modern UI. The transformation of the aging legacy system to an agile IT environment resulted in accelerated deployment of lightweight, low-maintenance applications, as well as auto-scaling, high availability and load balancing, data reconciliation and guaranteed message delivery – no more data loss.

The microservices were deployed in a RedHat OpenShift/Docker-based container platform and integrated with external services such as File Server, database and mail servers. A UI was developed to view and reconcile all Kafka messages. Rigorous functional and performance testing resulted in a highly configurable framework, with all pub-sub and transform services seamlessly integrated with Kafka and MQ.

Notes on Architecture:



Highlights

- Build SpringBoot based Apps
- Build Docker containerized apps
- Application deployed as Pods inside OpenShift platform
- Access service such as MQ, Kafka, Oracle as external services from within Openshift Pods

→ **Software components** included Docker 1.13.1, OpenShift Origin containers, v1.4.1 cloud platform.

→ **The deployment architecture** included Docker image-based implementation for micro-services using Java language and SpringBoot as framework component.

→ **Connectivity to backend services** like MQ and Oracle was provided from inside the containers and all services/routes exposed for service discovery between Docker/OpenShift.

SECTION VI
CONTAINERIZATION for C-LEVEL

"Let's do something with containers to speed up application delivery." If your company is like most today, infrastructure and operations leaders are hearing this from their C-level execs at every meeting. Clearly, the buzz around containers has reached boardrooms and convinced some business leaders to give the green light to do "something." But what do they expect? It's important to remember that containers are still mysterious to many decision makers outside IT, yet they need to understand the business benefits of each use case to provide support on the somewhat bumpy road ahead.

Containerization proposals must therefore include financial justification, and that's been difficult for some teams. In the spirit of transparency, a significant amount of refactoring is required before existing applications can take advantage of container architecture features, such as microservices. However, by calculating improvements in deployment speeds, cross-platform portability and delivery of end-user features and functionality along the way, your teams can present executive-level business cases that garner the support you need.

As every team soon discovers, containerization tools and technologies change almost daily, so working with an experienced technology partner like Xoriant may facilitate knowledge exchange and maximize your ROI on short or long-term initiatives.

REFERENCES

1 https://www.cio.com/article/3434010/more-enterprises-are-using-containers-here-s-why.html

2 https://www.projectmanager.com/ultimate-agile-guide/onboarding-to-agile-the-process-part

3 https://www.devopsdigest.com/2020-devops-containers-kubernetes-predictions-1

4 https://www.cio.com/article/2924995/what-are-containers-and-why-do-you-need-them.html

5 https://dzone.com/articles/why-enterprises-havent-adopted-containers-and-what

6,7 https://blog.kumina.nl/2017/04/the-benefits-of-containers-and-container-technology/; https://www.monitis.com/blog/top-5-benefits-of-containerization/

9 https://www.docker.com/resources/reports/idc-white-paper-rise-enterprise-container-platform

XORIANT

Xoriant is a Silicon Valley headquartered product engineering, software development and technology services firm with offices in the U.S., Europe and Asia. For both technology companies and enterprises, from startups to the Fortune 100, we leverage our expertise in emerging technologies and our high performing teams to deliver innovative solutions that accelerate time to market and keep our clients competitive.

Across all our technology focus areas – Product Engineering, DevOps, Cloud, Infrastructure & Security, Big Data & Analytics, Data Management & Governance, Digital and IoT – every solution we develop benefits from our product engineering pedigree. For 30 years and counting we have taken great pride in the long-lasting, deep relationships we have with our clients.

Learn more at www.xoriant.com.

Connect with us:

PE@xoriant.com +1 408 743 4400