Astadia™ THE MAINFRAME TO CLOUD EXPERTS

WHITE PAPER

Unisys ClearPath MCP & OS2200 Mainframes to Red Hat OpenShift on IBM Cloud Platform

Reference Architecture

Key Takeaways

- The UNISYS Mainframe Software Environment
- The options for Mainframe-To-Cloud Migrations
- The 5Rs - Replatform, Refactor, Replace, Rewrite, Retire
- The Power of Red Hat OpenShift
- IBM's Cloud Platform

Table of Contents

About this document

Introduction

Evolving customer behaviors and expectations are driving digital transformation across industries. Customers and partners often expect to interact with an organization, in real-time, whether at a physical location, through a website, using a mobile app, or by phone with a live customer service agent. The same is true with employees: they continue to demand efficient ways to connect with their organization and with each other.

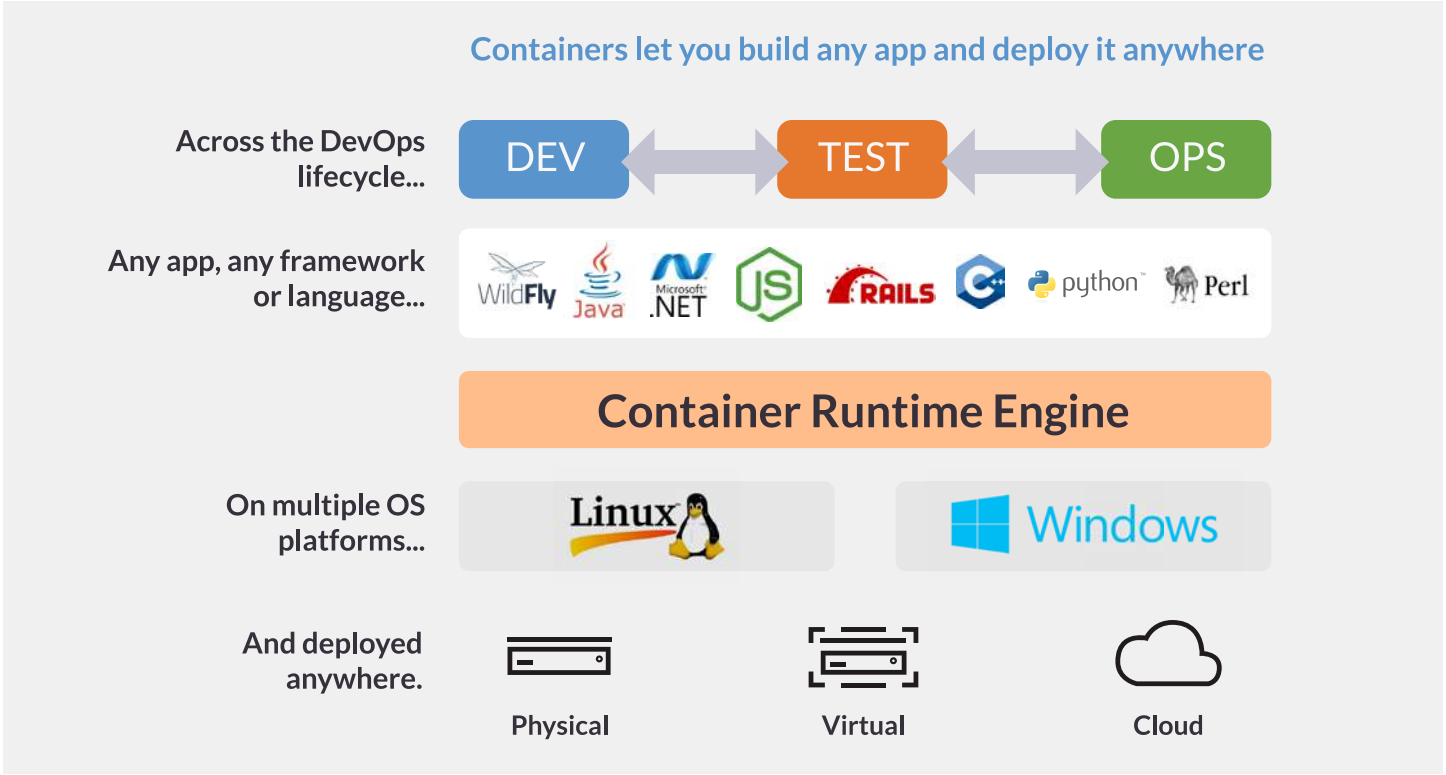
An organization's ability to engage users and maintain a competitive advantage depends in large part on the speed with which it can migrate or modernize its legacy mainframe software to cloud environments to enable user experiences. This "need for speed" has led to the rapid adoption of containerized application deployment—a key technology for boosting developer speed and productivity.

Containerization is about encapsulation and software management. It has broad appeal because any application that runs on Linux (and more recently, Windows) can be containerized. Developers can easily package an application and all its dependencies into a single image that can be promoted from development, to test, and to production—without change. Containers make it easy to ensure consistency across environments and multiple deployment targets like physical servers, virtual machines (VMs), and private and/or public clouds. This helps teams more easily develop and manage the applications that deliver real business value.

For applications: Containers make it easier for developers to build and promote an application and its dependencies as a unit. Containers can be deployed in seconds. In a containerized environment, the software build process is the stage in the life cycle where application code is integrated with needed runtime libraries.

For infrastructure: Containers represent sandboxed application processes on a shared Linux® OS kernel. They are more compact, lighter, and less complex than virtual machines and are portable across different environments—from on-premises to public cloud platforms.

Based on Kubernetes, Red Hat® OpenShift® is one of the most widely used open-source container orchestration platforms for enterprise application development and deployment. OpenShift on IBM® Cloud Platform is an excellent target environment for transitioning applications from Unisys® ClearPath® MCP (Burroughs®) or OS2200 (Sperry) mainframes to hybrid cloud implementations. With extensive security features and the flexibility to scale based on demand for the services, OpenShift on IBM Cloud® offers a complete operational environment in support of Unisys ClearPath MCP & OS2200 mainframe workloads that have been migrated to the cloud. OpenShift on IBM Cloud also supports innovation and evolution of the application portfolio in ways that were previously impossible due to the inflexible nature of the Unisys mainframe computing model, thus improving the productivity of application developers, support personnel, and end users alike.



Even more than a typical IT project, careful planning for modernization of Unisys mainframe applications and databases is the most important phase of the total project effort. A good place to begin is with a thorough assessment of the existing Unisys mainframe application portfolio. Through the assessment process, all aspects of the existing portfolio will be inventoried and examined in detail, resulting in a catalog of each application, database, technology platform and business user profile currently in use. Once completed, the results of this application rationalization will guide the sequence of application migration, as well as the different modernization strategies and techniques that may be called upon over the course of the entire project. We have included an overview of how Astadia® tackles legacy modernization projects with our Success Methodology to give you an idea of what is involved. In most cases, clients will begin the Assessment process with the most important business applications in the mainframe portfolio.

Do not let the enormity and importance of a Unisys ClearPath MCP or OS2200 mainframe migration project deter you from getting started, because the clock is ticking: Most organizations rely on programmers from the baby-boom generation to maintain legacy mainframe applications. Though they now represent just 25% of the workforce, these older workers may be the only employees with the skills to maintain legacy systems – often because they created them in the first place. But older workers are now retiring in droves, taking with them years or even decades of accumulated institutional knowledge of legacy applications and the underlying business rules that define them.

Millennials are now the largest portion of the U.S. labor force, representing 35% of all workers. Unfortunately, most have little to no understanding of legacy code or the mainframe environments in which it runs, because those skills are no longer being taught in today's computer science courses and trade schools. Younger graduates usually have little desire to learn COBOL and similar languages either, often viewing them as obsolete and not helpful to their career progress. Of even greater concern, younger workers typically lack knowledge of the underlying business rules upon which legacy applications are based. As those rules evolved over the years, mainframe systems were frequently updated by legacy workers without documentation. As these older employees retire, current IT teams may not even understand how the applications operate.

For organizations that depend on Unisys ClearPath MCP or OS2200 mainframes for mission-critical systems, the loss of legacy expertise creates a world of risks and potential problems, including operational inefficiencies and downtime along with damage to revenue, reputation, and brand. These potential risks are growing exponentially as more and more legacy coders log off for the last time. Meanwhile, hardware and software maintenance costs continue to escalate, and meeting the increasing demands of customers, employees and partners requires greater innovation than Unisys mainframe platforms can support.

How to Use This Reference Architecture Guide

We suggest that you begin by reading the Migrating Unisys ClearPath Mainframe Applications to OpenShift on IBM Cloud Platform section immediately below. After you have read that, here are our recommendations for which sections in this paper be read, based on role:

- Unisys Mainframe Experts: Skip to Understanding Red Hat OpenShift, continue with About IBM Cloud Platform, and conclude with Unisys Mainframes to OpenShift on IBM Cloud Reference Architecture.
- OpenShift & IBM Cloud Experts: Start with Understanding Unisys Mainframe Architecture, followed by Unisys Mainframes to OpenShift on IBM Cloud Reference Architecture.
- Business Leaders: To avoid drowning in technical detail, spend extra time to focus on the business benefits, approaches and challenges of mainframe migration in the sections immediately below; and then skip to the Ensuring Project Success section at the end of the document.

Migrating Unisys ClearPath Mainframe Applications to OpenShift on IBM Cloud Platform

Over the past ten years, cloud computing—both public and private—has emerged as the foundation of future enterprise technology. In terms of technology generations, mainframes are at least two and perhaps three generations old. Industry pundit claims that "the mainframe is finally dead" accompany every new wave of technology, but they have been wrong every time: Mainframes still exist today and continue to run key financial, operational, healthcare, government, military, and other vital and sensitive application systems around the world.

Containers have broad appeal because they allow users to easily package an application, and all its dependencies, into a single image that can be promoted from development, to test, and to production without change. Containers make it easy to ensure consistency across environments and multiple deployment targets like physical servers, virtual machines (VMs), and private and/or public clouds. This helps teams more easily develop and manage the applications that deliver business value.

So why should you migrate your Unisys ClearPath mainframe workloads; why migrate them to OpenShift on IBM Cloud Platform; and why is now the right time to do so?

Benefits of Mainframe Modernization

The specific benefits gained by migrating Unisys ClearPath MCP and OS2200 mainframe workloads will vary between organizations, and even at the application and database level. But in general, here are three of the top reasons driving Unisys legacy modernization projects:

Cost – The economics of OpenShift on IBM Cloud Platform computing are very compelling when compared with the maintaining the status quo of running a Unisys ClearPath MCP or OS2200 mainframe environment. A total-cost-of-ownership (TCO) evaluation of the subscription-based, consumption-driven cost model of the cloud, versus the exorbitant hardware and software maintenance costs of on-premises mainframes, will show a very appealing return-on-investment (ROI) that is achievable in the short-term—potentially less than 12 months from project completion.

People – As discussed earlier, the pool of available talent with relevant knowledge and experience in Unisys ClearPath MCP and OS2200 mainframe technology and programming languages is shrinking exponentially each year. Because Red Hat OpenShift on IBM Cloud Platform leverages skills already ingrained into younger software engineers, modernizing to these platforms and languages greatly increases the number of people who can work on the system. A full review of all existing code and applications is a key step in the migration process. By having both older and younger IT staff working together in this review, the institutional knowledge that would otherwise be lost as older workers retire should be recaptured to the benefit of the organization.

Flexibility – OpenShift on IBM Cloud Platform offers an open systems environment that facilitates high productivity and rapid innovation. A properly designed infrastructure scales easily and quickly, expanding and shrinking in sync with business demand. Backup, redundancy and disaster recovery is seamless. Support for multiple end-user platforms and devices is inherent. Database sharing across the enterprise with high performance is achievable, as is the ability to distribute the application and data geographically to keep it closer to the locations of users.

Approaches to Mainframe Modernization

You may notice that we use the terms "mainframe modernization" and "mainframe migration" throughout this document. Migration is but one type of modernization; modernization encompasses a broader set of strategies and options. In many cases, you will employ a combination of these strategies, with the right mix being determined during the critical application portfolio rationalization step of the project's assessment phase. Here are four of the most common approaches to modernization:

Replatform – Often called "Lift & Shift", this is a process that reuses the existing code, programs, and applications (typically written in COBOL) by moving them off the mainframe and recompiling the code without changes to run in a Unisys ClearPath mainframe emulator

hosted within an IBM Cloud instance. Since no modifications to the code are made, this approach minimizes the upfront risks and length of the project while most quickly delivering at least some hardware and software cost savings. Replatforming is most often used for standalone programs that have no dependencies or integrations with other systems. For less complex applications, the term "Rehosting" is often used interchangeably with "Replatforming." Rehosting is defined as moving applications to new platforms without recompiling any programs in the process.

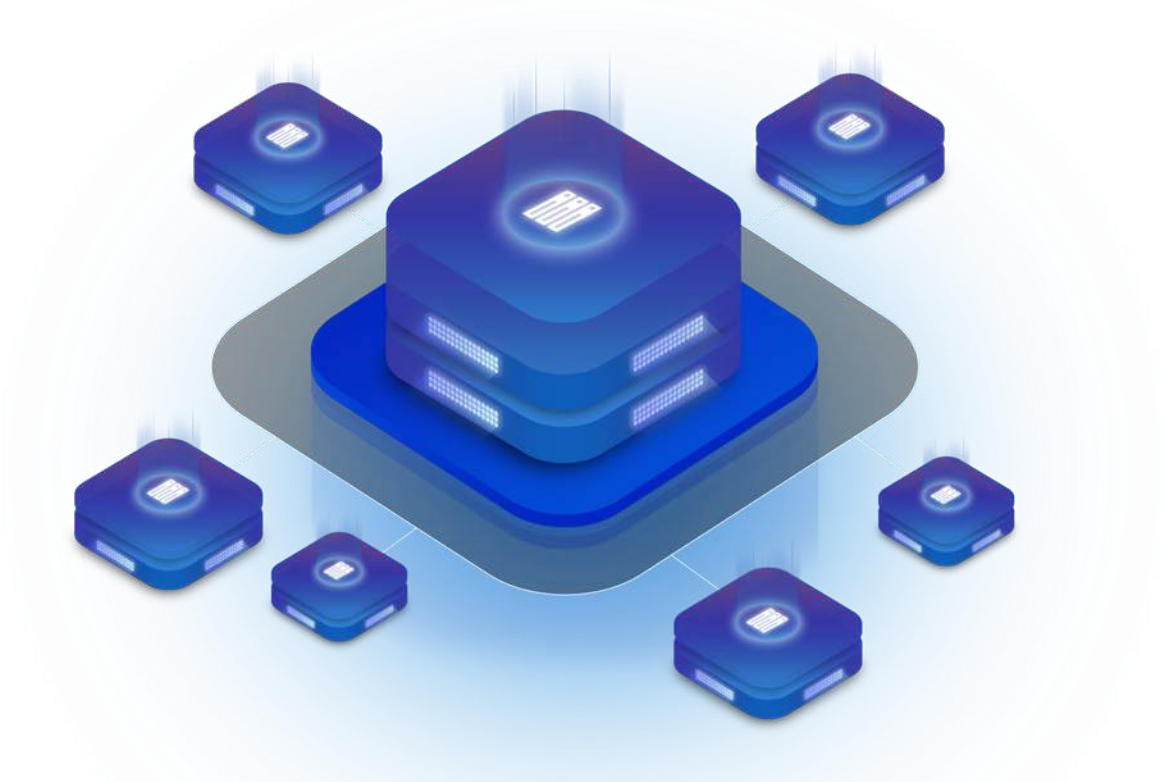
Refactor - In many situations, an optimal move to a cloud platform is achieved by transforming the existing code to object-oriented programming languages like Java or C#. There are many ways to accomplish this, but with complex mainframe applications involving millions of lines of code, the use of automated transformation tools is the only practical strategy. Manual rewrites are simply not realistic. (The term "rearchitecting" is sometimes used interchangeably with "refactoring.")

Rewrite – It may be tempting to say, "Let's just write new programs from scratch" to modernize Unisys ClearPath MCP and OS2200 mainframe applications. This approach is extremely risky and fails a vast majority of the time. It is complex, costly, and time consuming; and the resources and investment required tend to greatly exceed the forecast. A new, modern codebase may still be the correct end objective, but often a better approach would be to first move the applications to an OpenShift on IBM Cloud-based emulator, migrate the database to an IBM Cloud-based database, then focus on replacing modules/code over a planned, multi-phased approach. When it is time to rewrite, several automated code transformation engines are available to reduce the effort and minimize the risk.

Replace – Another approach to Unisys mainframe modernization is to completely replace the mainframe functionality with one or more off-the-shelf applications, typically delivered via a Software-as-a-Service (SaaS) model. We often see this with purpose-built solutions for finance, human resources, manufacturing, sales management, enterprise resource planning, etc. There are also industry-specific applications that may address business needs for which custom Unisys mainframe solutions were necessary decades ago, long before the widespread availability of COTS (Commercial-off-the-Shelf) packaged applications.

The upside of using SaaS is that your organization no longer worries about maintaining code. However, you will find that, while you can configure a SaaS application with various options provided by the vendor, you will not be able to truly customize your instance since the shared codebase will be run by all tenants (so-called Multi-Tenant), customers/organizations using the standard "service". This means that your current business processes may need to change to be compatible with those designed into the SaaS application, and not the other way around.

There are additional variations on these four modernization strategies, and you will likely employ a combination of several in achieving complete migration away from Unisys ClearPath MCP and OS2200 mainframes. It is a commonly accepted best practice among legacy modernization professionals to start with the lower-risk, lower-cost replatforming approach whenever possible to capture the gains and benefits of a cloud environment in the shortest time frame, followed by a deliberate and phased approach to refactoring, rewriting and/or replacing the applications.



Challenges of Mainframe Modernization

Mainframe migration projects are complex and require close management of the processes, budgets, and timelines that have been set as project goals. For example, replatforming is not simply a matter of moving source code from a Unisys ClearPath MCP or OS2200 mainframe to OpenShift on IBM Cloud Platform. It is likely that successful recompiling of that code will also require some re-engineering and refactoring. The project will also involve data and file conversions for transitioning the databases to the IBM Cloud, either manually or aided by third-party Extract/Transform/Load (ETL) tools.

As we have emphasized, the first challenge of any Unisys mainframe modernization project is to develop a rock-solid plan, based upon a thorough application portfolio assessment and rationalization. As you put your plan together and begin to execute, here are some additional factors you will need to consider:

Documentation

Many large and complex application portfolios in Unisys mainframe environments do not have adequate documentation detailing exactly what these applications do, and how they do it. Many applications have become maintenance nightmares: they are decades old and have likely been modified and enhanced many times over the years, often with no documentation of the changes. The external interactions with these systems – their inputs and outputs – may be the only things known to the business about these older applications. With the original developers having often retired or left the company, the internal program logic and processing is now a black box that everyone hopes will just keep working.

Migrating a minimally documented system of this nature is tricky, and thorough testing prior to a "go live" decision is critical to mitigating this issue. (And of course, copious documentation should be captured during the migration process for the resulting system.)

Application-Specific Challenges

A couple of general points about the application portfolio should be noted. As mentioned above, a lack of detailed documentation about these aging systems makes the migration effort more difficult. The project teams driving these types of migration projects must resort to detailed analysis of the actual application source code to reverse-engineer the exact behavior of the application.

Another important application-specific issue is developing an understanding of the integration requirements and dependencies of the application with other systems and databases. These integrations and dependencies must be clearly identified and, if still needed, re-connected (or possibly rebuilt) and made operational along with the migrated system. In many cases, this work will require "hand-crafting".

Running Parallel Systems

For some period, there may need to be some parallel processing between the Unisys mainframe application while it is still being used in production, and the newly migrated version running on the OpenShift on IBM Cloud platform. Planning and executing this parallel processing will be a challenge, and will require extra time and coordination to make it successful.

Organizations may also choose to run parallel systems to achieve quick reductions in Unisys mainframe processing cycles by moving both the development and test environments to OpenShift on IBM Cloud, while keeping the production system on the Unisys ClearPath MCP or OS2200 mainframe for the interim period.

Data Integrity

Moving the contents of large databases is very challenging on several levels. Typically, a database "cleanup" will be necessary to ensure that the content of the new target database is as accurate and complete as possible.

A Unisys ClearPath MCP and OS2200 mainframe modernization project is a good time to transform, correct, and validate the organization's data.

Speed to Completion

In almost every project, speed will be a top priority. In fact, speed is part of the reason for the growing popularity of DevOps (or DevSecOps, which integrates security more prominently), and is also one of the drivers for containerization of workloads. The costs and complexities of extended project cycles can have an enormous negative impact in both tangible and intangible terms. As project cycles are extended, staff attrition and fatigue can become major factors in causing slower progress vs. the original plan.

Paying for continued operation of the primary production system while simultaneously funding development efforts for the new system

will have a temporary negative financial impact for as long as that duality continues. Thus, getting the new system to a "go live" status quickly and efficiently and retiring the old system will accelerate ROI while keeping unexpected costs to a minimum. At the same time, however, it is important that development be done incrementally to minimize the chances of failure.

Project Funding

It is very important for any modernization project to be properly funded and supported by the business management teams and executives. This support is essential to maintain project continuity and sufficient funding throughout the project cycle. As stated earlier, speed will be a key factor in the project execution so funding must be in place to sustain that speed.

Expertise

Unisys mainframe migration projects come in many forms. But in every case, a variety of specialist skills will be needed on the project team. These specialists may include business analysts working to reverse-engineer and understand the business rules reflected within the legacy application code.

The team will also require experts in specific programming languages, databases, networks, terminal devices and many other components of the total application portfolio that will need to be addressed over the course of the migration to OpenShift on IBM Cloud. Staff must be available to address any specific functionality or use case of the Unisys ClearPath MCP or OS2200 mainframe environment.

All these technical facets of the application must be replicated by functionality on the target OpenShift on IBM Cloud and work as they did in the original Unisys ClearPath MCP and OS2200 mainframe environment. Thorough testing by the project team, followed with testing by the business users of the original mainframe application system, is an absolute requirement to validate functional equivalence between the legacy and new systems. Once testing is completed, a final performance and tuning (P&T) exercise will ensure that the new OpenShift on IBM Cloud Platform deployment is performing at optimal levels.

Understanding Unisys Mainframe Architecture

General-purpose mainframes have served as the backbone of business computing since they were developed in the late 1940s. For more than fifty years, each mainframe manufacturer continuously enhanced their unique architectures to outperform competitors and meet evolving business demands. IBM and Unisys eventually dominated the market and became the gold standards of mainframe computing.

Unisys Mainframe Heritage

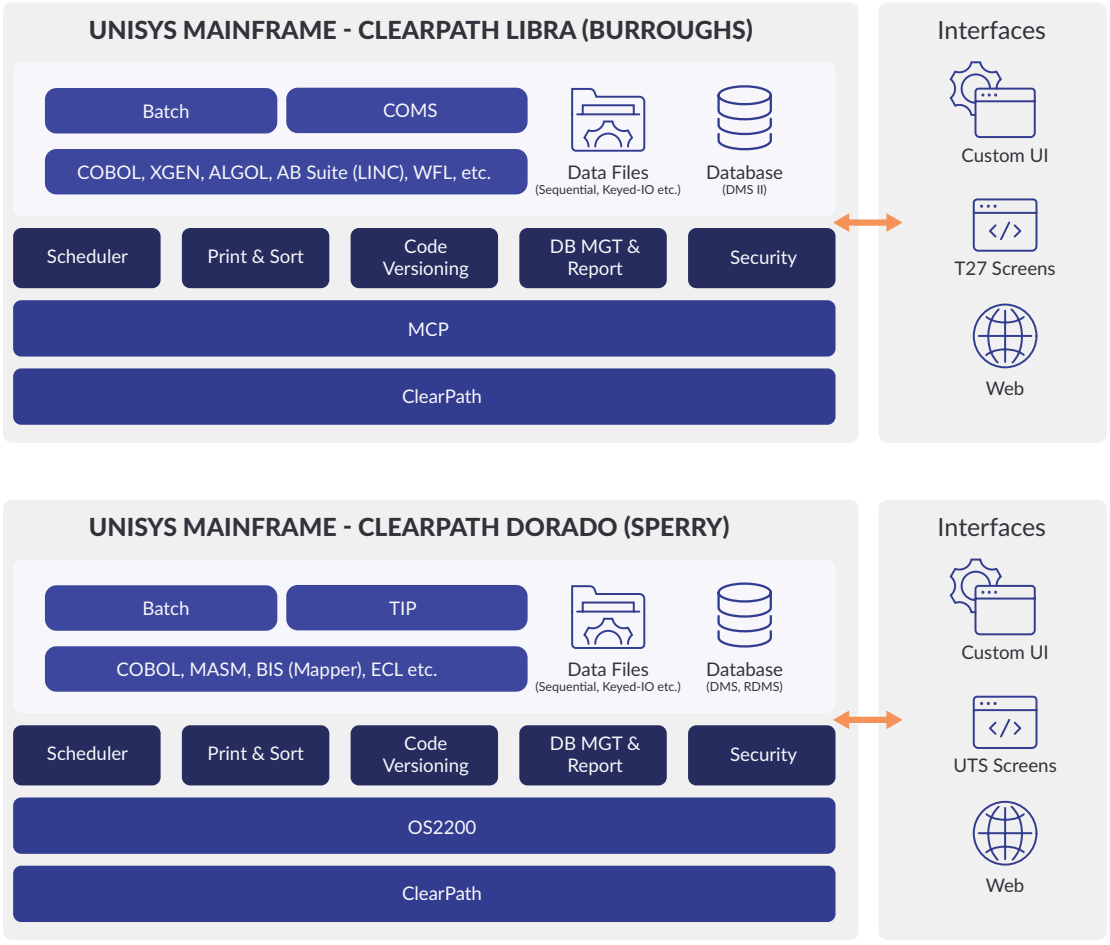
Unisys can trace its roots all the way back to 1886 and the American Arithmometer Company, which later became the Burroughs Corporation. Best known for its adding machines, Burroughs entered the digital computer market in 1953 and became a leading supplier of mainframes and other related products for the finance and banking industries.

Meanwhile, Sperry Corporation was founded in 1910 as the Sperry Gyroscope Company. In 1955, Sperry merged with Remington Rand, which had developed the first general-purpose computers known as BINAC and UNIVAC. Sperry's Univac division was first called Sperry Rand, and later became Sperry Univac.

In 1978, Sperry sold off all its non-computer divisions and went back to its original Sperry Corporation name.

The Unisys Corporation of today was formed in 1986, when Burroughs merged with Sperry to become Unisys. At the time of the merger, each company had its own line of mainframe computers, each with a loyal customer base. There were attempts to unify the two architectures and their respective technologies, but the two distinct systems survive to this day.

Unisys ClearPath Libra mainframes originated from the Burroughs line, while ClearPath Dorado's heritage is Sperry. Even though there are distinct differences between these two sets of mainframe technologies, they are in many ways very similar and share many of the same basic characteristics.



Unisys Mainframe Components

As discussed earlier, migration of Unisys ClearPath MCP and OS2200 mainframe applications to new architectures like Red Hat OpenShift on IBM Cloud involves far more than a simple "lift and shift" of the application's source code. Every aspect of an application's current mainframe operating environment must be considered and accommodated within the target cloud environment. Here are a few common examples:

USER INTERFACES

Users access the mainframe applications through a variety of means. Often, they work with green screen terminal emulators that provide a character mode interface (T27 and UTS).

Alternatively, custom user interfaces can be built on top of the character mode interface to provide a more user-friendly and locked-down interface to specific mainframe applications. For example, a web-based or mobile application within a kiosk, equipped with a card scanner and touchscreen display, could serve as a self-serve front-end to the mainframe for banking customers.

BATCH

All mainframes provide batch environments that handle bulk data processing workloads. Jobs are submitted to the system (WFL or EFL), scheduled (often during overnight or other off-peak times when online processing loads are minimal), and executed without further operator interaction. Output from batch jobs is spooled, printed and distributed to users.

TRANSACTION PROCESSING

Transaction processing is at the core of most mission-critical applications, with thousands or millions of transactions being processed daily. Mainframes provide the online processing environments (COMS, TIP) that make this possible. Security, transaction integrity, and predictably fast response times are of particular importance for this type of workload.

PROGRAMMING LANGUAGES

Mainframes offer a wide assortment of programming languages to meet customer needs. Most business applications are written in COBOL, but other languages are also used when better suited to the application's needs. Algol, MASM, BIS/MAPPER, FORTRAN, Pascal, PL/I, and 4GL development products like XGEN and ABSuite (LINC/EAE) are also used for Unisys application development.

DATA FILES

Mainframes store data in files with different record organizations and media types. Data files can be sequential, direct access, fixed or variable length, blocked or unblocked, etc. Data files may be stored on disks, magnetic tapes, CDROMs, etc. For the most part, data in these files are stored in Extended Binary Coded Decimal Interchange Code (EBCDIC), an eight-bit character encoding system used primarily on mainframes.

DATABASES

Mainframes provide high performance database management systems to support mission-critical online applications. These databases may be offered by the hardware vendor or by a third party, and can feature hierarchical (DMSII), networked (DMS), or relational (RDMS) structures. Some have proprietary data manipulation languages; others rely on open standards such as SQL; but all are designed to deliver high levels of availability, integrity, consistency, reliability, security, and auditing. Database software makes intensive use of the computing and input/output capabilities of the mainframe to provide optimal response times.

ENVIRONMENTAL SOFTWARE

Mainframes also run software to support the management, operation, application development, and overall security of the system. Unisys and third parties cooperate to provide the environmental software required to create a robust but controlled environment for customer applications.

Scheduling software is used to manage the execution of workflow streams. Output management systems handle the collection, storage and distribution of reports to the users that require them. Source code management systems are used to maintain application source code by tracking versions, as well as release lifecycles.

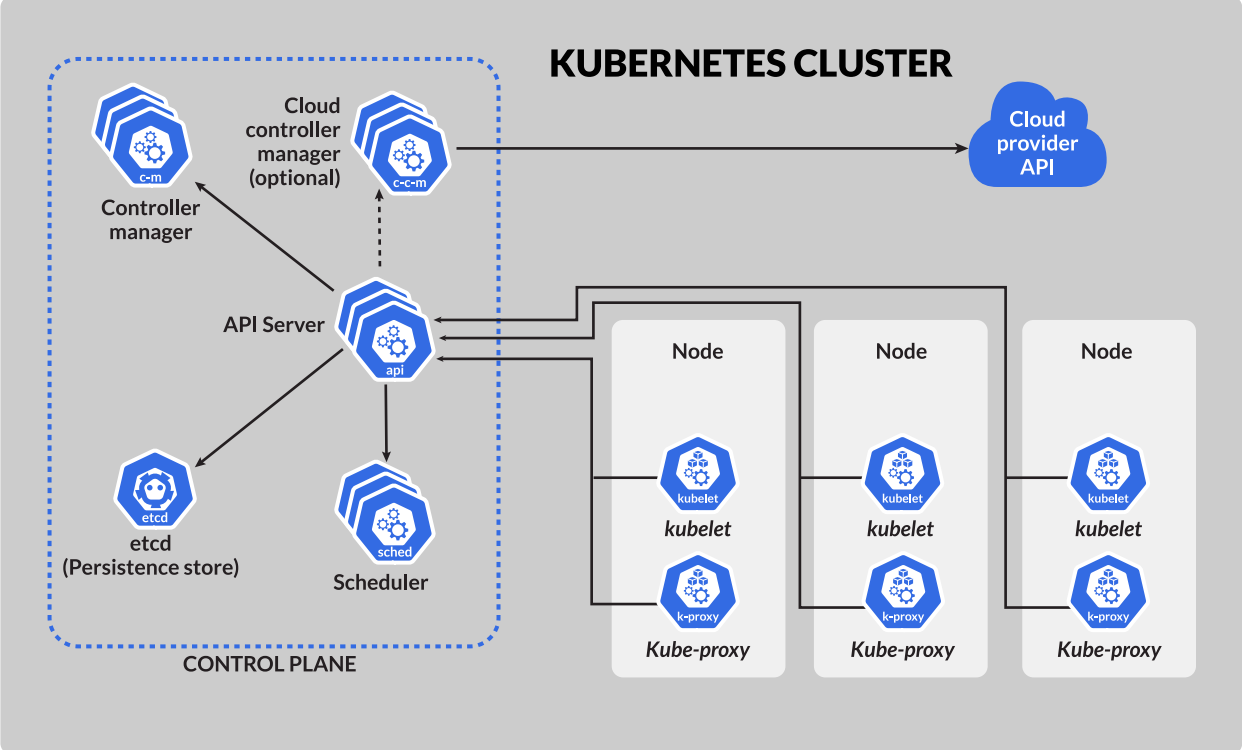
Security is tightly controlled at all levels of the mainframe environment by security software designed to minimize the risk of data exposure and ensure regulatory compliance.

Understanding Red Hat OpenShift

Overview

Today's businesses differentiate by delivering extraordinary customer experiences through applications that quickly evolve to meet their needs. Once deployed, these applications must be portable, highly secure, easy to scale, and simple to manage. Organizations are increasingly turning to containers and Kubernetes® to meet these needs.

Red Hat OpenShift includes everything needed for hybrid cloud, edge, enterprise container, and Kubernetes development and deployments. It includes an enterprise-grade Linux® operating system, container runtime, networking, monitoring, container registry, authentication, and authorization solutions. These components are tested together for unified operations on a complete Kubernetes platform spanning every cloud.



What is Kubernetes?

Kubernetes (also known as "k8s" or "kube") is an open-source container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications in the cloud. In other words, groups of hosts running workloads Linux containers can be clustered together, and Kubernetes helps to easily and efficiently manage those clusters.

Kubernetes clusters can span hosts across on-premises, public, private, or hybrid clouds, making it an ideal platform for hosting cloud-native applications that require rapid scaling.

Kubernetes was originally developed and designed by engineers at Google®. Google was one of the early contributors to Linux container technology and talks publicly about how everything at Google runs in containers. (In fact, this is the technology behind Google's own cloud services.) Google donated the Kubernetes project to the newly formed Cloud Native Computing Foundation (CNCF) in 2015.

What can you do with Kubernetes?

The primary advantage of using Kubernetes, especially for organizations optimizing applications development for the cloud, is that it provides the platform needed to schedule and run container clusters on physical or virtual machines (VMs.) Kubernetes runs "pods" consisting of one or more containers, with the atomic definition of a run-unit being a pod.

More broadly, Kubernetes helps fully implement a container-based infrastructure in production environments. And because Kubernetes is all about automation of operational tasks, it can do many of the same things other application platforms or management systems do—but for containers.

KUBERNETES ENABLES:

- Orchestrating containers across multiple hosts
- Minimizing the hardware resources needed to run enterprise apps
- Controlling and automating application deployments and updates
- Mounting and adding storage to run stateful apps
- Scaling containerized applications and their resources on-the-fly
- Declaratively managing services, which guarantees deployed applications are always running the way they were intended to run

Using Kubernetes in Production with Red Hat OpenShift

Although it is possible to run and use straight Kubernetes, additional components are needed to create a production-grade platform including registry, authentication, networking, security and compliance, telemetry, monitoring, logs management, services, and other tools.

Also, Kubernetes is open source and as such, there is no formalized support structure around the base technology—at least not one that is responsive, dependable, and knowledgeable enough to trust for key business applications. If an organization has an issue with its implementation of Kubernetes while running in production, it would likely be frustrated trying to find a solution internally—and its customers would be as well.

That's where Red Hat OpenShift comes in.

OpenShift can be viewed as "Kubernetes ++"—it is a complete container platform built on fundamental building blocks of the Kubernetes foundation. OpenShift complements and enhances Kubernetes by removing the complexity of deploying and managing containers at the enterprise level, and also adds the pieces of missing technology needed to make it a complete and viable solution for the enterprise.

OpenShift enables Enterprise Kubernetes, and Red Hat has been a field leader since the first inceptions of OpenShift almost ten years ago. From day one, containerization built into the Linux kernel has been Red Hat's driving force in creating a flexible, secure, and self-serving platform.

OpenShift is the foundation for hybrid cloud deployments. It provides an abstraction API that allows the same code, deployment and configuration to exist unchanged whether running on-premises or in the cloud. This eliminates "vendor lock-in", because OpenShift owns every infrastructure interface – not the cloud provider. Applications running in OpenShift do not interact directly with the cloud platform; but only with the OpenShift/Kubernetes API.

Increasing Productivity with OpenShift

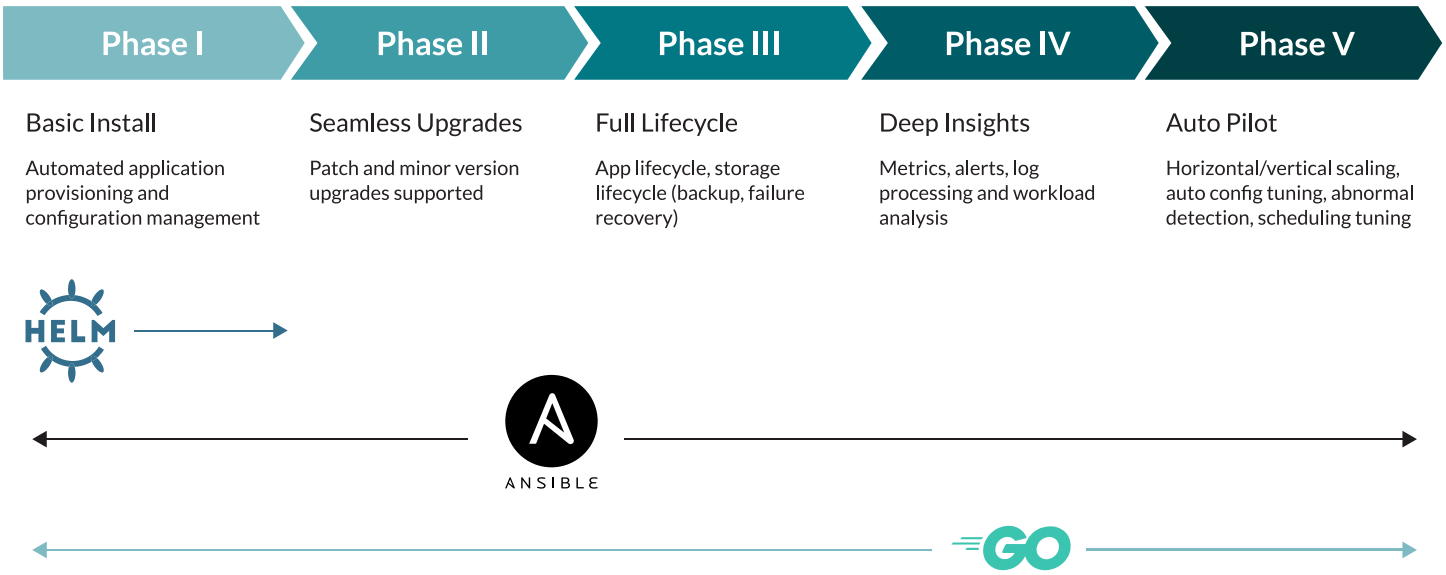
SIMPLIFY AND OPTIMIZE THE CODE COMPONENT

Many traditional code features like cluster state and load balancing are part of the OpenShift platform. Advanced features like Blue-Green and Canary deployments are all implemented as configuration settings and do not impact the code deployment. From legacy monolithic to modern micro-services code bases, OpenShift has ways to improve and optimize each of them:

- **Enables automated workflows**: S2I (Source to image), CI/CD pipelines, code validation, automatic updates, rollbacks and more.
- **Minimizes the need to know Kubernetes details on the developer side.** A wide range of developer tool sets, IDE integration makes using/deployment/managing runtimes on OpenShift a matter of a click or simple command.
- **Integrates into existing cloud providers.** By running OpenShift in AWS or Azure, OpenShift will implement features like Storage to take advantage of each cloud platform's strengths without aiming for the lowest common denominator.

ADDITIONAL ADVANCED FEATURES AND CAPABILITIES:

- **Service Mesh** provides communication among microservices-based applications using APIs
- **Serverless** enables applications to not consume resources unless needed. Instead, it will be automatically started (and stopped) based on need.
- **Pipelines** provides an advanced workflow to manage a code deployment from source to complex deployments across multiple clouds, dependencies etc. Full CI/CD chains can be created to automate the whole process including unit and integration testing from the moment a developer commits code to a source repository.



AUTOMATING THE FULL COMPONENT LIFECYCLE WITH OPERATORS

With OpenShift 4, Red Hat introduced "operators" into Kubernetes —a concept which has now been adopted by CNCF (Cloud Native Computing Foundation) as a standard component of Kubernetes. Operators provide an open and easy to use interface to automate the full lifecycle of a component, from initial installation to the "auto pilot" production phase.

An Operator consists of a set of container images, along with a set of files describing the various entry points into those container images. Each entry point is called at a specific lifecycle stage of the application. Operators perform tasks which are usually performed by humans such as upgrading a clustered application to a new version, resizing, backing up, and so on.

The foundation of operators is a new type of Kubernetes object called a CRD (Custom Resource Definition). A CRD, which can only be added by administrators, defines terms (kinds) and role-based access controls (RBACs) to control who/what can use these terms. The Kubernetes API server then implements the CRD by creating new REST paths. Put simply, an operator is a packaged combination of a CRD, a custom controller, and domain specific knowledge.

By encapsulating domain expertise, operators enable internal developers and third-party vendors alike to make their solutions easily consumable as services. To enable users to quickly locate specific operators, Red Hat created an online catalog of community and certified third-party operators called the Operator Hub (operatorhub.io). OpenShift administrators can choose from this large marketplace of operators and, with a single click, have an operator and associated code installed directly into their cluster. Once installed, upgrades and changes to the operator across multiple clusters are handled by OpenShift's Operator Lifecycle Manager (OLM), including over-the-air updates from Red Hat and other partner vendors.

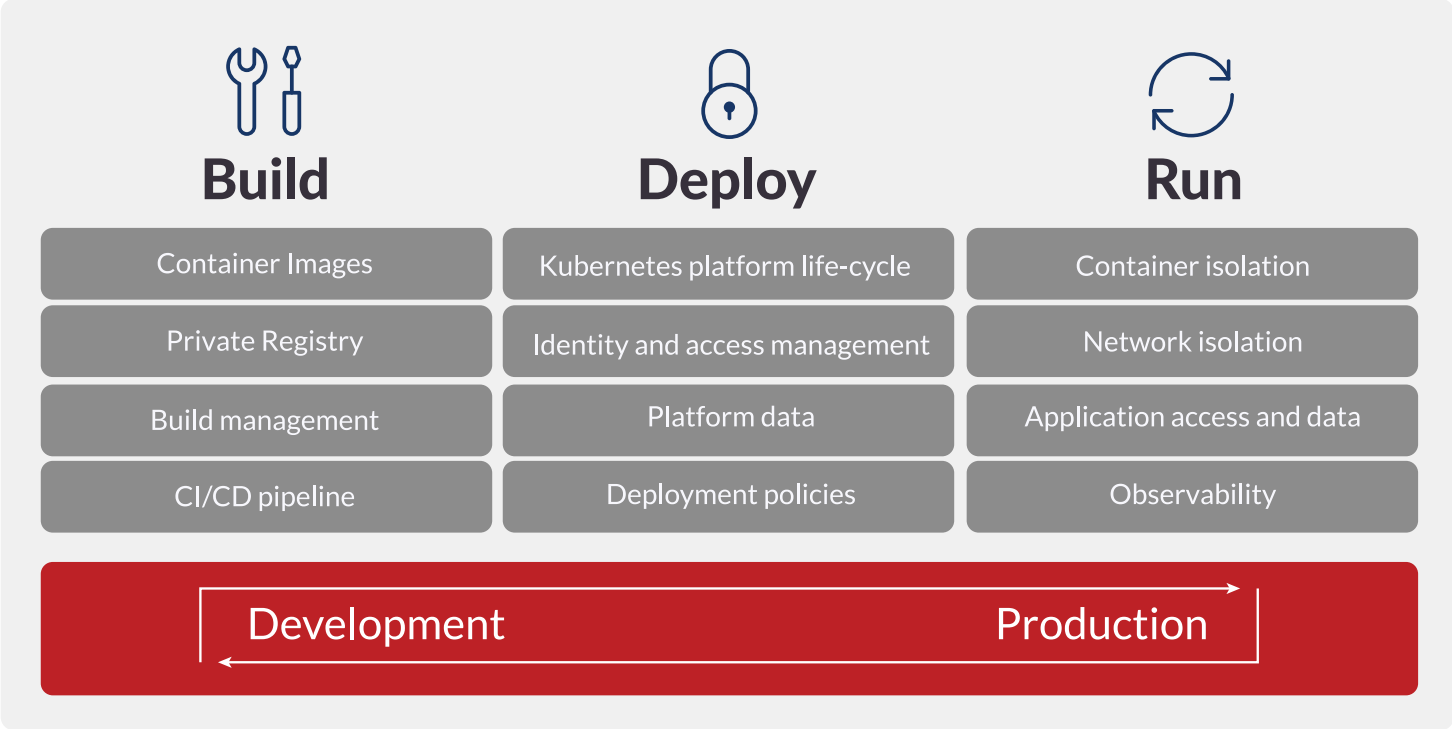
Operators improve productivity by automating and abstracting system operation tasks. End users do not have to understand the inner workings of the technology they install, and ongoing maintenance is defined and controlled by the owner of the operator —whether it's Red Hat, IBM, or another certified partner. Any issue with an operator can be reported back to the vendor, which can often diagnose, resolve, and automatically distribute an update before customers even notice. Without operators, owners providing services would have to field far more support calls involving very hard-to-diagnose problems – problems which could not be solved without deep domain-specific knowledge by Red Hat Support, the partner, and the customer alike.

PRIMARY OPENSHIFT FEATURES AND BENEFITS:

PLATFORM FEATURE	BENEFIT
Scalability	Workloads designed for dynamic scaling can scale to thousands of instances across hundreds of nodes, automatically without any operator intervention.
Multi-cluster federation	Consolidated views of clusters
Persistent storage	OpenShift Container Native Storage (CNS) allows for full integration into a wide range of vendor and Red Hat-supported storage backend solutions, without requiring the code to know the implementation differences. Although not the only persistent storage solution, CNS is recommended for hybrid deployments because it provides the same interface regardless of where the cloud deployment is done. End users are able to request complex storage without needing admin privileges.
Container portability	OpenShift uses Open Containers Initiative (OCI) industry standards for all container images and run-time. This means containers built by other OCI-compliant container tools will work without modification.
Self-service	Developers (the users of OpenShift) can quickly and easily create applications on demand, allocate resources and implement security. The system can be pre-provisioned with a set of standard settings, so developers do not have to do the same job every time a new project is created.
Multi-language	Any programming language and paradigm existing and working on Linux is available. Windows native containers are now also supported, allowing for C# native code to be run as part of OpenShift.
User interfaces	A rich Web portal accessible through many different types of devices, a broad set of command-line tools, and integration into traditional IDE solutions and even built-in Web based IDEs.
Automation	OpenShift comes with several CI/CD frameworks in addition to the automation built into the platform. Complex workflows can be designed/executed on the platform to respond to events. Streamlined and automated container and application builds, deployments, scaling, health management and more.
Virtualization	Run existing virtual machines as you would a container. This provides a single interface and knowledge base for the users of OpenShift.
Automated installation and upgrades	Automated installation and over-the-air platform upgrades are supported in the cloud with IBM Cloud, Amazon Web Services, Google Cloud Platform, and Azure; and on-premises using vSphere, OpenStack, Red Hat Virtualization, or bare metal.
Robust ecosystem	An ever-expanding ecosystem of vendors supported by the introduction of operators provides a wide variety of integrations and features ready to install on top of OpenShift. This includes storage, network, CI, ISV, security, and more.

Comprehensive container and Kubernetes Security

With many organizations now running essential services on containers, ensuring container security and compliance has become critical—from build to deploy to run. Securing containers is a lot like securing any other running process: Developers need to think about security across all layers of the solution stack before deploying and running a container, as well as throughout the application and container life cycle. Importantly, security must be viewed as a continuous process, extending to respond to new threats and solutions as they emerge. A comprehensive approach to container security involves securing the build, deployment, and runtime; and extending security with specialized tools when appropriate.



Containers make it easier for developers to build and promote an application and its dependencies as a unit. Containers also make it easy to extract maximum value from servers by enabling multi-tenant application deployments on a shared host, spinning up and shutting down individual containers as needed. Unlike traditional virtualization, you do not need a hypervisor to manage guest operating systems on each VM. Containers virtualize your application processes, not your hardware.

Of course, applications are rarely delivered in a single container. Even simple applications typically have a frontend, a backend, and a database. And deploying modern microservices-based applications in containers means deploying multiple containers—sometimes on the same host, and sometimes in a complex environment distributed across multiple hosts or nodes.

There are many considerations when managing container deployment at scale, such as:

- Which containers should be deployed to which hosts?
- Which hosts have more available capacity?
- Which containers need access to others, and how will they discover one another?
- How should access to and management of shared resources such as network and storage be controlled?
- How should container health be monitored?
- How can application capacity be scaled automatically to meet demand?
- How can developer self-service be enabled while also meeting security requirements?

A custom container management environment can be constructed, but requires a considerable investment of time to identify, integrate and manage individual components. A better approach is to deploy a container platform with built-in management and security features. This lets your team focus on building the applications that provide business value rather than needlessly reinventing infrastructure.

Red Hat OpenShift Container Platform delivers a consistent hybrid cloud enterprise Kubernetes platform for building and scaling containerized applications. Organization-wide use of Kubernetes requires additional security capabilities to help developers build security into applications, automate policies for managing container deployment security, and to protect the container runtime.

Building security into applications

Building security into applications is critical for cloud-native deployments. To secure containerized applications, it is important to:

- Use only trusted container content
- Employ an enterprise container registry
- Control and automate the building of container images
- Integrate security into the application pipeline

1. Use only trusted container content

When managing security, what is inside the container matters. For some time now, applications and infrastructures have been composed from readily available components. Many of these are open-source packages, such as the Linux operating system, Apache Web Server, Red Hat JBoss® Enterprise Application Platform, PostgreSQL, and Node.js. Containerized versions of these packages are now also available. However, as with any code downloaded from an external source, it is important to know where the packages originated, who built them, and whether they contain any malicious code. Some questions to ask:

- Will the container contents compromise my infrastructure?
- Are there known vulnerabilities in the application layer?
- Are the runtime and OS layers in the container up to date?
- How frequently will the container be updated and how will we know when it is updated?

Red Hat has been packaging and delivering trusted Linux content for years in Red Hat Enterprise Linux and is now delivering that same trusted content packaged as Linux containers. With the introduction of Red Hat Universal Base Images, the greater reliability, security, and performance of Red Hat container images can be leveraged wherever Open Container Initiative (OCI)-compliant Linux containers run. This means a containerized application can be built on a Red Hat Universal Base Image, pushed to the container registry of choice, and then shared.

Red Hat also provides certified images and operators for various language runtimes, middleware, databases, and more via the Red Hat Ecosystem Catalog. Red Hat certified containers and operators run anywhere Red Hat Enterprise Linux runs, from bare metal servers to VMs to cloud, and are supported by Red Hat and its partners.

Red Hat continuously monitors the health of the images it delivers. The Container Health Index exposes the "grade" of each container image, based in part on the age and impact of unapplied security errata to all components of a container and providing an aggregate rating of container safety that can be understood by security experts and nonexperts alike.

Whenever Red Hat releases security updates, any affected container images are also rebuilt and pushed to the public registry. Red Hat security advisories alert users to any newly discovered issues in certified container images and provide links to updated images so that organizations can, in turn, update any applications that use the image.

There may be times when content is needed that Red Hat does not provide. Container scanning tools based on continuously updated vulnerability databases should be used to ensure that the latest information on known vulnerabilities is available when using container images from other sources. Because the list of known vulnerabilities is constantly evolving, the contents of container images should be checked when initially downloaded, and vulnerability status should continue to be tracked over time for all approved and deployed images.

2. Employ an enterprise container registry

Development teams build containers that layer content on top of downloaded public container images. Access to, and promotion of, both downloaded and internally-built container images should be secured in the same way as other types of binaries. Several private registries support storage of container images, and it is recommended that a private registry be selected that helps automate policies for the use of container images stored in the registry.

OpenShift includes a private registry that provides basic functionality to manage container images. This registry uses role-based access control (RBAC) to manage who can pull and push specific container images. OpenShift also supports integration with other private registries, such as JFrog's Artifactory or Sonatype Nexus. A private or "golden registry" should be deployed and shared globally within the organization or agency, typically using a separate non-OpenShift cluster.

Red Hat Quay is also available as a standalone enterprise registry, offering many additional enterprise features such as geographic replication and build image triggers.

The Clair project is an open-source engine that powers the Red Hat Quay security scanner to detect vulnerabilities in all images stored within Quay. The OpenShift Container Security Operator integrates with Quay to provide a cluster-wide view of known vulnerabilities for deployed images in the OpenShift console.

3. Control and automate the building of container images

Managing the container build process is key to securing the software stack. Adhering to a "build once, deploy everywhere" philosophy ensures that the result of the build process is exactly what is then deployed in production. It is also important to maintain the immutability of containers. In other words, running containers should not be patched, but instead rebuilt and redeployed. OpenShift provides several capabilities for automating builds based on external events to improve the security of custom images.

- Red Hat Quay triggers provide a mechanism for spawning a repository build of a Dockerfile from an external event such as a GitHub push, BitBucket push, GitLab push, or webhook.
- Source-to-Image (S2I) is an open-source framework for combining source code and base images. Using S2I, development and operations teams can easily collaborate on a reproducible build environment. When a developer commits code with Git under S2I, OpenShift can:

Trigger automatic assembly of a new image from available artifacts, including the S2I base image and the newly committed code (via webhooks on the code repository or some other automated CI process).

Automatically deploy the newly built image for testing.

Promote the tested image to production status and automatically deploy the new image through the continuous integration and deployment (CI/CD) process.

- OpenShift image streams can be used to watch changes to external images deployed in a cluster. Image streams work with all the native resources available in OpenShift, such as builds or deployments, jobs, replication controllers, or replica sets. By watching an image stream, builds and deployments can receive notifications when new images are added or modified and react by automatically launching a build or deployment, respectively.

4. Integrate security into the application pipeline

OpenShift includes integrated instances of Tekton, a next-generation Kubernetes CI/CD pipeline that works for containers (including serverless), as well as the older Jenkins for CI. Rich RESTful APIs are also provided to integrate other build or CI/CD tools including private image registries, so organizations are free to use the build tools provided in OpenShift or rely on an external workflow of their choice.

A best practice for application security is to integrate automated security testing into the pipeline, including the registry, integrated development environment (IDE), and CI/CD tools.

REGISTRY:

Container images can and should be scanned in the private container registry. Red Hat Quay with the Clair security scanner can be used to notify developers when vulnerabilities are discovered. The OpenShift Container Security Operator integrates with Red Hat Quay to provide a cluster-wide view in the OpenShift console of known vulnerabilities for deployed images. Alternatively, multiple third-party certified container scanning solutions can be found in the Red Hat Ecosystem Catalog.

IDE:

Red Hat Dependency Analytics integrated development environment (IDE) plugins provide vulnerability warnings and remediation advice for project dependencies when the code is first brought into the IDE.

CI/CD:

Scanners can be integrated with CI for real-time checking against known vulnerabilities that catalog the open-source packages in a container, notify of any known vulnerabilities, and provide updates when new vulnerabilities are discovered in previously scanned packages.

Additionally, the CI process should include policies that flag builds with issues discovered by security scans so teams can take appropriate action to address those issues as soon as possible.

Finally, signing of custom-built containers is recommended to ensure they are not tampered with between build and deployment.

Managing configuration, security, and compliance of a deployment

Effective security of a deployment includes securing the Kubernetes platform as well as automating deployment policies. OpenShift includes the following capabilities out of the box:

- Platform configuration and life cycle management
- Identity and access management
- Securing platform data and attached storage
- Automating policy-based deployment

5. Platform configuration and life cycle management

The Cloud Native Computing Foundation (CNCF) Kubernetes Security Audit, published in summer 2019, concluded that the greatest security threat to Kubernetes is the complexity of configuring and hardening Kubernetes components. Red Hat OpenShift meets that challenge through Kubernetes Operators.

An Operator is a method of packaging, deploying, and managing a Kubernetes-native application. An Operator acts as a custom controller that can extend the Kubernetes application programming interface (API) with the application-specific logic required to manage the application. Every OpenShift component is wrapped in an operator, delivering automated configuration, monitoring, and management for OpenShift. Individual operators directly configure components such as the API server and the software-defined network (SDN), while the cluster version operator manages multiple operators across the platform. Operators automate cluster management, including updates, from the kernel to services higher in the stack.

One of the key values of a container platform is that it enables developer self-service, making it easier and faster for development teams to deliver applications built on approved layers. A self-service portal gives teams enough control to foster collaboration while still providing security. The Operator Lifecycle Manager (OLM) provides the framework for OpenShift cluster users to find and use operators to deploy the services needed to enable their applications. With OLM, users can install, upgrade, and assign role-based access control to available operators.

To help with compliance, OpenShift provides the Compliance Operator to automate the platform's conformity with technical controls required by compliance frameworks. The Compliance Operator lets OpenShift administrators describe the desired compliance state of a cluster. It then provides them with an overview of any compliance gaps along with approaches to remediate them. The Compliance Operator assesses conformity of all platform layers, including the nodes running the cluster. The File Integrity Operator is also available to run regular file integrity checks on the cluster nodes.

6. Identity and access management

Given the wealth of features in Kubernetes for both developers and administrators, strong identity management and RBAC is a critical element of the container platform. The Kubernetes APIs are key to automating container management at scale. For example, APIs are used to initiate and validate requests, including configuring and deploying pods and services.

API authentication and authorization are critical for securing a container platform. The API server is a central point of access and should receive the highest level of security scrutiny. The OpenShift control plane includes built-in authentication through the Cluster Authentication operator.

Developers, administrators, and service accounts obtain OAuth access tokens to authenticate themselves to the API. Administrators can configure the identity provider of choice to the cluster so users can authenticate before receiving a token. Nine identity providers are currently supported, including lightweight directory access protocol (LDAP) directories.

Fine-grained RBAC is enabled by default in Red Hat OpenShift. RBAC objects determine whether a user is allowed to perform a given action within a cluster. Cluster administrators can use the cluster roles and bindings to control access levels to the OpenShift cluster and to projects within the cluster.

7. Securing platform data and attached storage

Red Hat OpenShift hardens Kubernetes by default to secure data in transit. It also includes options for securing data at rest.

- OpenShift protects platform data in transit by:
- Encrypting data in transit via https for all container platform components communicating with each other.
- Sending all communication with the control plane over transport layer security (TLS).
- Ensuring all access to the API Server is X.509 certificates or token-based.
- Using project quota to limit the damage a rogue token could inflict.
- Configuring etcd with its own certificate authority (CA) and certificates. (In Kubernetes, etcd stores the persistent master state while other components watch etcd for changes to bring themselves into the specified state.)
- Rotating platform certificates automatically.

OpenShift protects platform data at rest by:

- Optionally encrypting Red Hat Enterprise Linux CoreOS disks and the etcd datastore for additional security.
- Providing Federal Information Processing Standards (FIPS) readiness for Red Hat OpenShift. FIPS 140-2 is a U.S. government security standard used to approve cryptographic modules. When Red Hat Enterprise Linux (RHEL) CoreOS is booted in FIPS mode, OpenShift platform components call RHEL cryptographic modules. Clusters become FIPS compliant, and non-compliant workloads are not allowed to run.

Containers are useful for both stateless and stateful applications. Red Hat OpenShift supports both ephemeral and persistent storage. Protecting attached storage is a key element of securing stateful services. Red Hat OpenShift supports multiple storage types, including network file system (NFS), Amazon Web Services (AWS) Elastic Block Stores (EBS), Google Compute Engine (GCE) Persistent Disks, Azure Disk, iSCSI, and Cinder.

In addition, OpenShift Container Storage is persistent software-defined storage integrated with and optimized for the OpenShift Container Platform. OpenShift Container Storage offers highly scalable, persistent storage for cloud-native applications that require features such as encryption, replication, and availability across the hybrid multi-cloud.

Of course, the security features available in any chosen storage solution should also be used.

8. Automating policy-based deployment

Strong security includes automated policies that can be used to manage container and cluster deployment from a security point of view.

POLICY-BASED CONTAINER DEPLOYMENT

Red Hat OpenShift clusters can be configured to allow or disallow images to be pulled from specific image registries. It is a best practice for production clusters to only allow images to be deployed from your private registry.

OpenShift's Security Context Constraints (SCC) admission controller plugin defines a set of conditions that a pod must run with in order to be accepted into the system. Security context constraints let you drop privileges by default, which is important and still the best practice. OpenShift SCCs ensure that, by default, no privileged containers run on OpenShift worker nodes. Access to the host network and host process IDs are also denied by default.

Users with the required permissions can adjust the default SCC policies to be more permissive if they choose.

Red Hat Advanced Cluster Management for Kubernetes provides advanced application lifecycle management using open standards to deploy applications using placement policies that are integrated into existing CI/CD pipelines and governance controls.

POLICY-BASED MULTI-CLUSTER MANAGEMENT

Deploying multiple clusters can be useful to provide high availability for applications across multiple availability zones or functionality. Multiple clusters can also be used for common management of deployments or migrations across multiple cloud providers, such as IBM Cloud, Amazon Web Services (AWS), Google Cloud, and Microsoft Azure. When managing multiple clusters, orchestration tools will need to provide the security required across the different deployed instances. As always, configuration, authentication, and authorization are key – as is the ability to pass data securely to applications, wherever they run, and managing application policies across clusters. Red Hat Advanced Cluster Management for Kubernetes provides:

- Multi-cluster lifecycle management that allows creating, updating, and destroying Kubernetes clusters reliably, consistently, and at scale.
- Policy-driven governance risk and compliance utilizes policies to automatically configure and maintain consistency of security controls according to industry corporate standards. Compliance policies can be specified to apply across one or more managed clusters.

Protecting running applications

Beyond infrastructure, maintaining application security is equally critical. Securing containerized applications requires:

- Container isolation
- Application and network isolation
- Securing application access
- Observability

9. Container isolation

To take full advantage of container packaging and orchestration technology, the operations team needs the right environment for running containers. Operation teams need an operating system (OS) that can secure containers at the boundaries: securing the host kernel from container escapes and securing containers from each other.

Containers are Linux processes with isolation and resource confinement that enable running sandboxed applications on a shared host kernel. The method to secure containers should be the same as the method to secure any running process on Linux.

NIST special publication 800-190 recommends using a container-optimized OS for additional security. As the operating system base for OpenShift, Red Hat Enterprise Linux (RHEL) CoreOS reduces the attack surface by minimizing the host environment and tuning it for containers. RHEL CoreOS only contains the packages necessary to run OpenShift, and its userspace is read-only. The platform is tested, versioned, and shipped with OpenShift and is managed by the cluster. RHEL CoreOS installation and updates are automated and always compatible with the cluster. It also supports the infrastructure chosen by the organization, inheriting most of the RHEL ecosystem.

Every Linux container running on a OpenShift platform is protected by powerful RHEL security features built into OpenShift nodes. Standard Linux process security features such as namespaces, SELinux, Cgroups, capabilities, and secure computing mode (seccomp) are also used to secure containers running on RHEL.

Linux namespaces provide the fundamentals of container isolation. A namespace makes it appear to the processes within the namespace that they have their own instance of global resources. Namespaces provide the abstraction that gives the impression of an application running on its own operating system from inside a container.

SELinux provides an additional layer of security to keep containers isolated from each other and from the host. SELinux allows administrators to enforce mandatory access controls (MAC) for every user, application, process, and file. SELinux is like a brick wall that will stop you if you manage to break out of the namespace abstraction, either accidentally or on purpose. SELinux mitigates container runtime vulnerabilities, and well-configured SELinux configurations can prevent container processes from escaping their containment.

Cgroups (control groups) limit, account for, and isolate the resource usage (e.g., CPU, memory, disk I/O, network) of a collection of processes. Cgroups prevent container resources from being stomped on by another container on the same host. They can also be used to control pseudo devices – a popular attack vector.

Linux capabilities can be used to lock down privileges in a container. Capabilities are distinct units of privilege that can be independently enabled or disabled. When running containers, multiple capabilities can be dropped without impacting most containerized applications.

Finally, a seccomp (secure computing mode) profile can be associated with a container to restrict available system calls.

10. Application and network isolation

Multi-tenant security is essential for enterprise-scale use of Kubernetes. Multi-tenancy allows different teams to use the same cluster while preventing unauthorized access to one another's environments. OpenShift supports multi-tenancy through a combination of kernel namespaces, SELinux, RBAC, Kubernetes (project) namespaces, and network policies.

OpenShift projects are Kubernetes namespaces with SELinux annotations. Projects isolate applications across teams, groups, and departments. Local roles and bindings are used to control who has access to individual projects.

Security context constraints let you drop privileges by default, which is important and still the best practice. Red Hat OpenShift security context constraints (SCCs) ensure that, by default, no privileged containers run on OpenShift worker nodes. Access to the host network and host process IDs are denied by default.

Deploying modern microservices-based applications in containers often means deploying multiple containers distributed across multiple nodes. These microservices need to discover and communicate with each other. With network defense in mind, a container platform is needed that can take a single cluster and segment the traffic to isolate different users, teams, applications, and environments within that cluster. Tools are also needed to manage inbound external access to the cluster, as well as outbound access from cluster services to external components. Achieving network isolation requires the following key properties:

Ingress traffic control: OpenShift includes CoreDNS to provide a name resolution service to pods. The OpenShift router (HAProxy) supports ingress and routes to provide external access to services running on-cluster. Both support reencrypt and passthrough policies.

Network namespaces: The first line in network defenses comes from network namespaces. Each collection of containers (known as a "pod") gets its own IP and port range to bind to, thereby isolating pod networks from each other on the node. The pod IP addresses are independent of the physical network that OpenShift nodes are connected to.

Network policies: The OpenShift SDN uses network policies to provide fine-grained control of communication between pods. Network traffic can be controlled to any pod from any other pod, on specific ports and in specific directions. When network policies are configured in multi-tenant mode, each project gets its own virtual network ID, thereby isolating project networks from each other on the node. In multi-tenant mode (by default) pods within a project can communicate with each other but pods from different namespaces cannot send packets to or receive packets from pods or services of a different project.

Egress traffic control: Red Hat OpenShift also provides the ability to control egress traffic from services running on the cluster using either router or firewall methods. For example, IP whitelisting can be used to provide access to an external database.

11. Securing application access

Securing your applications includes managing application user and API authentication and authorization.

CONTROLLING USER ACCESS

Web single sign-on (SSO) capabilities are a key part of modern applications. Container platforms can come with several containerized services for developers to use when building their applications. Red Hat SSO is a fully supported, out-of-the-box security assertion markup language (SAML) 2.0 or OpenID Connect-based authentication, web single sign-on, and federation service based on the upstream Keycloak project. Red Hat SSO features client adapters for Red Hat Fuse and Red Hat JBoss Enterprise Application Platform. Red Hat SSO enables authentication and web single sign-on for Node.js applications and can be integrated with LDAP-based directory services including Microsoft Active Directory and RHEL Identity Management. Red Hat SSO also integrates with social login providers such as Facebook, Google, and Twitter.

CONTROLLING API ACCESS

APIs are key to applications composed of microservices. These applications have multiple independent API services, leading to proliferation of service endpoints which require additional tools for governance. We recommend using an API management tool. Red Hat 3scale API Management gives you a variety of standard options for API authentication and security that can be used alone or in combination to issue credentials and control access.

The access control features available in Red Hat 3scale API Management go beyond basic security and authentication. Application and account plans let you restrict access to specific endpoints, methods, and services, and apply access policies for groups of users. Application plans allow you to set rate limits for API usage and control traffic flow for groups of developers. You can set per-period limits for incoming API calls to protect your infrastructure and keep traffic flowing smoothly. You can also automatically trigger overage alerts for applications that reach or exceed rate limits and define behavior for over-limit applications.

SECURING APPLICATION TRAFFIC

Securing application traffic with cluster ingress and egress options is covered in section 10 of this paper. For microservice-based applications, security traffic between services on the cluster is equally important. A service mesh can be used to deliver this management layer. The term "service mesh" describes the network of microservices that make up applications in a distributed microservice architecture and the interactions between those microservices.

Based on the open source Istio project, OpenShift Service Mesh adds a transparent layer on existing distributed applications for managing service-to-service communication without requiring any changes to the service code. OpenShift Service Mesh uses a multi-tenant operator to manage the control plane life cycle, enabling OpenShift Service Mesh to be used on a per-project basis. Furthermore, OpenShift Service Mesh does not require cluster-scoped RBAC resources.

OpenShift Service Mesh provides discovery, load balancing, and most importantly, service-to-service authentication and encryption, failure recovery, metrics, and monitoring.

12. Observability

The ability to monitor and audit an OpenShift cluster is an important part of safeguarding the cluster and its users against inappropriate usage. OpenShift includes built-in monitoring and auditing, as well as an optional logging stack.

OpenShift Container Platform services connect to the built-in monitoring solution composed of Prometheus and its ecosystem. An alert dashboard is available. Cluster administrators can optionally enable monitoring for user-defined projects. Applications deployed to OpenShift can be configured to take advantage of the cluster monitoring components.

Auditing events is a security best practice and is generally required to comply with regulatory frameworks. At its core, OpenShift auditing was designed using a cloud-native approach to provide both centralization and resiliency. In OpenShift, host auditing and event auditing are enabled by default on all nodes. OpenShift provides extraordinary flexibility for configuring management and access to auditing data. The amount of information that is logged to the API

server audit logs can be controlled by choosing which audit log policy profile to use.

Monitoring, audit, and log data is RBAC-protected. Project data is available to project administrators and cluster data is available to cluster administrators.

As a best practice, clusters should be configured to forward all audit and log events to a security information and event management (SIEM) system for integrity management, retention, and analysis. Cluster administrators can deploy cluster logging to aggregate all the logs from the OpenShift cluster, such as host and API audit logs, as well as application container logs and infrastructure logs. Cluster logging aggregates these logs from throughout the cluster nodes and stores them in a default log store. Multiple options are available for forwarding logs to the SIEM of choice.

Extending security with a robust ecosystem

To further enhance container and Kubernetes security or to meet existing policies, organizations may choose to integrate with third-party security tools. Red Hat supports a broad ecosystem of certified partners offering solutions such as:

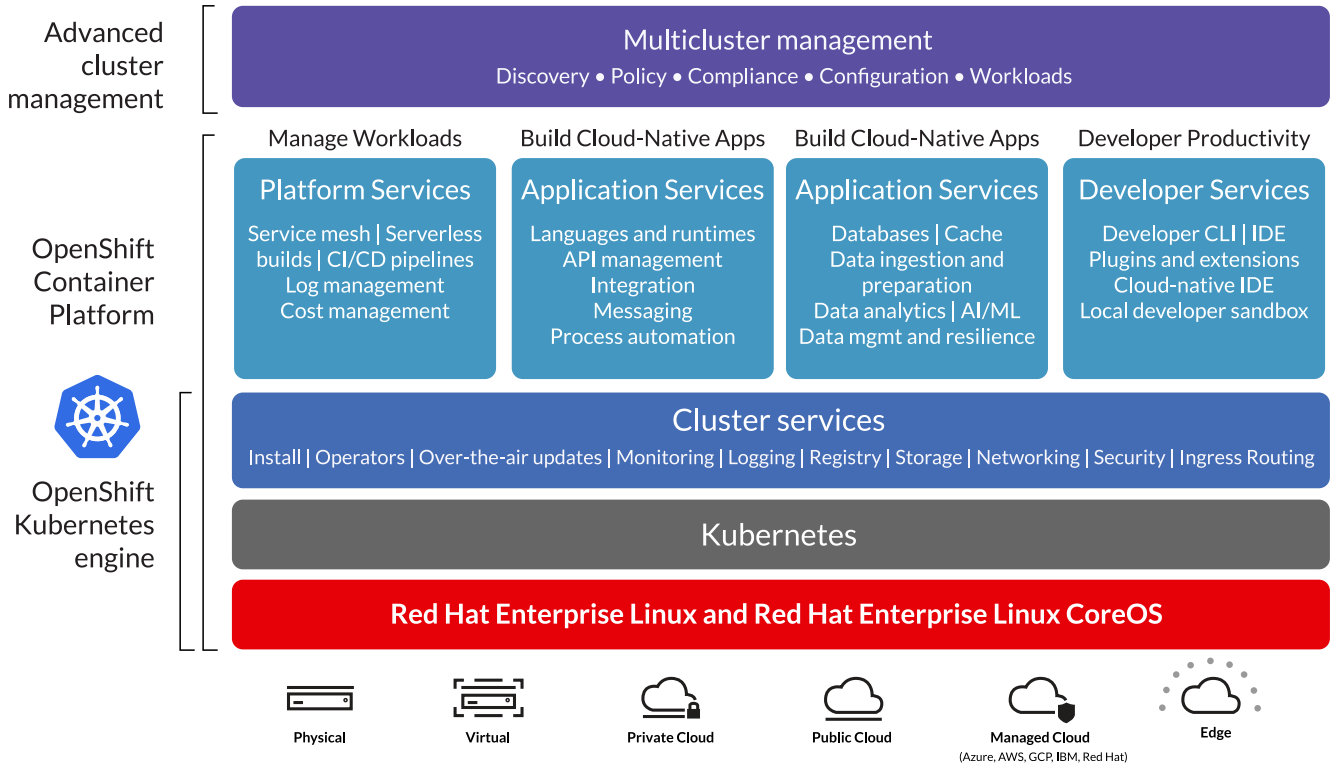
- Privileged access management
- External certificate authorities
- External vaults and key management solutions
- Container content scanners and vulnerability management tools
- Container runtime analysis tools
- SIEM

Summary

Deploying container-based applications and microservices is not only about security. A container platform also needs to provide an experience that empowers both developers and operators without compromising the functions needed by each team, while also improving operational efficiency and infrastructure utilization.

OpenShift is built on a core of standard and portable Linux containers that incorporates a complete range of built-in security and compliance features, including:

- Integrated build and CI/CD tools for secure DevOps practices.
- Hardened, enterprise-ready Kubernetes with built-in platform configuration, compliance, and lifecycle management.
- Strong RBAC with integrations to enterprise authentication systems.
- Options for managing cluster ingress and egress.
- Integrated SDN and service mesh with support for network microsegmentation.
- Support for securing remote storage volumes.
- Red Hat Enterprise Linux CoreOS, optimized for running containers at scale with strong isolation.
- Deployment policies to automate runtime security.
- A single switch set during installation makes a cluster FIPS-compliant and prevents non-compliant workloads from running.
- Integrated monitoring, audit, and logging.
- A vast ecosystem of partners that provide additional compliance verification capabilities and handling of container-based workloads.



OpenShift provides a full featured technology stack across physical virtual, private, and public clouds.

About IBM Cloud Platform

IBM Cloud Platform provides the technology and services that make running Unisys ClearPath MCP and OS2200 mainframe applications in the cloud a safe, secure, reliable way to achieve high performance results and enable future innovation for the organization.

OpenShift on IBM Cloud provides a broad set of infrastructure services, such as computing power, storage options, networking and databases that are delivered as a utility: on-demand, available in seconds, with pay-as-you-go pricing. From data warehousing to deployment tools, directories to content delivery, several OpenShift on IBM Cloud Platform services are available.

New services can be provisioned quickly, without upfront capital expense. This allows enterprises, start-ups, small and medium-sized businesses, and customers in the public sector to access the building blocks they need to respond quickly to changing business requirements.

IBM Cloud serves millions of active customers in more than 190 countries. IBM is steadily expanding global infrastructure to help customers achieve shorter latency times and higher throughput, to ensure that their data resides only in the region they specify.

As customers grow their businesses, IBM Cloud will continue to deliver infrastructure that meets their global requirements.

IBM Cloud Platform infrastructure is built around Regions and Availability Zones (AZs). A Region is a physical location in the world where IBM has multiple AZs. AZs consist of one or more discrete data centers, each with redundant power, networking, and connectivity, housed in separate facilities. These AZs offer clients the ability to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center. IBM Cloud Platform operates multiple AZs within several geographic Regions around the world, with more Availability Zones and more Regions coming online in the coming year.

Each IBM Cloud Region is designed to be completely isolated from the other IBM Cloud Regions. This achieves the greatest possible fault tolerance and stability. Each AZ is isolated, but the AZs in a Region are connected through low-latency links.

OpenShift on IBM Cloud Platform provides clients with the flexibility to place instances and store data within multiple geographic Regions as well as across multiple Availability Zones within each Region. Each Availability Zone is designed as an independent failure zone. This means that Availability Zones are physically separated within a typical metropolitan region and are in lower risk flood plains (specific flood zone categorization varies by Region).

In addition to discrete uninterruptable power supply (UPS) and onsite backup generation facilities, they are each fed via different grids from independent utilities to further reduce single points of failure. AZs are all redundantly connected to multiple tier-1 transit-providers.

Unisys Mainframes to OpenShift on IBM Cloud Reference Architecture

Virtual Private Clouds

The Virtual Private Cloud (VPC) enables provisioning a logically isolated section of the IBM Cloud where IBM Cloud resources are launched and managed in a virtual network that you define. The VPC is your private area within the IBM Cloud Platform.

The VPC can be thought of as the fence around all the systems that you have migrated into the IBM Cloud. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can use both IPv4 and IPv6 within your VPC for secure and easy access to resources and applications.

Computing Resources

OpenShift on IBM Cloud Platform provides secure, resizable compute capacity in the IBM Cloud. It serves as the foundation upon which an application sits. It is the container that holds the operating systems, mainframe emulators, application executables, and other supporting software that make up an application.

Depending on your specific circumstances, you may choose to separate some pieces into individual instances or run everything in a single instance. For example, you might have a section dedicated to batch COBOL and another dedicated to Online. Sections might even be segregated by application.

Storage

IBM Cloud mass storage can be thought of as a huge hard drive for virtually unlimited amounts of data. It serves as the primary virtual storage device for all instances running your migrated applications. The IBM Cloud also offers a low-cost, reliable service for backup and archiving of all types of data. These services combine to meet the storage requirements of your migrated Unisys ClearPath MCP and OS2200 mainframe applications.

Databases

IBM's Cloud SQL Relational Database Service is where all legacy relational data will reside. This includes any flat file data that has been converted to a relational database, such as Unisys flat files; as well as DMS, DMS II or RDMS data that has been converted to a normalized relational database design and migrated to IBM's Cloud SQL Relational Database Service.

This service is optimized for high database performance. It is cost-efficient, has resizable capacity, and is designed to reduce time-consuming database administration tasks.

IBM Cloud Platform includes support for several widely used database engines, including DB2®, Microsoft® SQL Server®, PostgreSQL® and MySQL®.

An analysis of your existing legacy databases and applications will reveal all the changes required to migrate data to IBM's Cloud SQL Relational Database Service, or any other RDBMS running on the IBM Cloud Platform.

Load Balancing

Applications with high transaction volumes need a way to balance the workload. The IBM Cloud Platform automatically distributes incoming application traffic across multiple instances to achieve scalability, high-performance, and fault tolerance in migrated applications. It provides the load balancing capability needed to route traffic evenly among multiple applications and keep them performing efficiently.

Security

In the OpenShift on IBM Cloud environment, Lightweight Directory Access Protocol (LDAP) will be used for accessing and maintaining distributed directory information services. While there are other possibilities, this is most likely where your legacy application user IDs, passwords, permissions, etc. will be captured and managed.

Hosting LDAP services on a smaller separate instance often makes it easier to maintain independently of applications. However, a full analysis of your legacy security environment will be required to determine how to best architect and configure security in the migrated system.

Monitoring

Every IT system needs to be monitored. The IBM Cloud Platform uses a monitoring service for cloud resources now running the legacy applications deployed to the IBM Cloud.

This tool collects and tracks metrics, monitors log files, sets alarms, and automatically reacts to changes in your IBM Cloud resources. This data is used to resolve problems quickly and keep your migrated applications running smoothly — much like you do on Unisys ClearPath MCP and OS2200 mainframes today. Other cloud-ready monitoring tools are available from 3rd parties as well.

Source Control

Just as you have products and processes to control your application sources and manage application releases on your Unisys mainframe today, you need a similar set of tools in the IBM Cloud.

IBM Cloud has a fully managed source control service providing secure and private Git repositories. It eliminates the need to operate your own source control system or worry about scaling its infrastructure.

This facility is where migrated application source code and binaries will be stored, together with new source, binaries, and anything else that should be archived.

Emulating Unisys Mainframe OS Features within OpenShift

As a container runtime platform, OpenShift provides a method to run and manage container workloads. It does not, however, decide what those workloads should be, what services (if any) they should provide, etc. Developers can add frameworks to a deployment as they see fit, or split code over multiple deployments based solely on functionality needs.

This means there are no core OS features available by default. However, these can be implemented and included within the containers as needed. An example is print (spool) services: a typical Linux/Unix deployment would have a running daemon listening for print-jobs, and when received would translate and queue the job to be delivered to a destination printer. A typical container would not include this daemon, but it could be added if such a feature is needed.

Again, a thorough analysis of your current Unisys application portfolio will identify the specific mainframe features that must be emulated within OpenShift containers during migration.

Mapping Unisys Terminology to OpenShift Equivalents

Core Architectural Concepts of OCP

Below are some core architectural concepts in OCP (OpenShift Container Platform):

CONCEPT	DESCRIPTION
Container	A process isolated by the kernel to a very limited view of the system. The process often only sees its own resources and uses no shared resources with anything else running on the same server.
Container image	The binary format of a container. This image is layered—one layer may be the base system, another the application platform and another the actual application. This allows for shared image components. Images are found in container repositories—OpenShift comes with its own internal registry, and Red Hat Quay or other private registries can be added to support advanced container registry options.
Pod	This is the smallest (atomic) unit in OpenShift. A pod has a single identity, single IP address and resources (memory, CPU, etc.). Inside a pod is at least one container. If more than one container exists within a pod, those containers can share memory space directly – in other words, the containers within a pod are not isolated from one another. However, pods themselves are isolated by default, and only the exposed ports (service) can be used to communicate to/from the pod.
Service	The internal exposure points of a pod. Typically using UDP/TCP ports, a network name is provided to allow other runtimes to find and address the pod. This will show up as DNS to any other application, so no special features are needed—the service name is all that is needed (the "hostname".) Services provide a load-balancing feature when more than one instance of a pod is running.
Deployment	A deployment (or Deployment Configurations as OpenShift used in the past) describes a desired state of a pod. Providing one or more container image references, OpenShift will use this definition to ensure that the desired number of instances are running at any point in time. This means if one pod fails, another one is automatically started. Deployments also provide the means to do health-checks on existing pods. Availability of a pod is not defined by a process, but on a given response being returned in a timely manner from the pod (is it alive/ready?).

Route	To allow outside access to code, a route exposes a service to the outside by providing it a URL (hostname) that is resolvable by users outside of OpenShift. An ingress router provides a reverse proxy mechanism for routing traffic from the outside to the right pod.
RBAC	Role Based Access Control allows project administrators (and cluster administrators) to delegate rights to cluster operations across the cluster and projects. These rights can be very granular or use the full cluster scope.
Project	Every component of OpenShift is deployed to a project—also known as a namespace in Kubernetes. By default, a user with access to one namespace cannot view objects in other namespaces unless specifically granted. Namespaces are not directly related to network security.
Network Policies	Every project owner can define ingress traffic rules, specifying which other namespaces may send traffic to it. A full multi-tenant network provides network isolation by default—only granted access is allowed between namespaces. For example, even within a namespace, a project owner can inject limitations ensuring that only an application can access a database.
Operator	Automation and custom PRDs allowing a custom API endpoint to automate a series of actions from a single call. This could be to install a new database, add a cluster feature like central logging etc. The OpenShift operatorhub.io site holds certified and community versions of operators available to install.
Compute Nodes	Compute nodes are the execution platform for pods. Every pod must run on a node to be running. As pods are ephemeral which node does not matter, and often even the developer/system-operator will not know (nor need to know) which node a given pod is currently running on. A SDN provides a logical cluster-wide network, tying workloads from multiple systems together. When a node fails, any runtimes on this node will be moved to other available nodes.
Master Nodes	A special kind of node that holds the OpenShift/Kubernetes control plane. This is where cluster state is kept and controlled, and where the main API end point resides. OpenShift requires three master nodes to provide quorum in case of a node failure.
PVC	The Physical Volume Claim is the mechanism by which a developer requests persistent block storage. The PVC will be received by a storage class that defines the interaction with a backend storage system. Multiple storage classes can exist, providing a wide range of options for storage. Once a PVC is complete, the developer associates the PVC in the deployment definition and specifies where this volume should be mounted in the container.
Secrets	Storing of configuration data like usernames and passwords cannot be done in clear text, and definitely not as hard-coded values. Secrets is a means to provide encrypted and restricted access to values that should not be shared. Only the project owner can view/change a secret's content, but any developer can mount/digest the data in the secret within their code. For example, a typical development scenario includes a development project, QA project and production project. Each project has a secret with the same name, but different values provide access to different systems depending on which environment the execution happens in. This makes the exact same deployment work and act differently.
Configuration Map	Values that are not considered sensitive can be provided using a configuration map. This is a value pair storage and can hold nested or embedded objects like images and certificates. A configuration map makes it simple to provide a managed source of all configuration data of any deployment.

Deploying Applications with OpenShift

Any basic deployment onto OpenShift will contain pieces of each of the components listed above. Other options exist but would be dependent upon the frameworks chosen and how the development integration is expected to work.

While this may sound complex, OpenShift makes it easy to use:

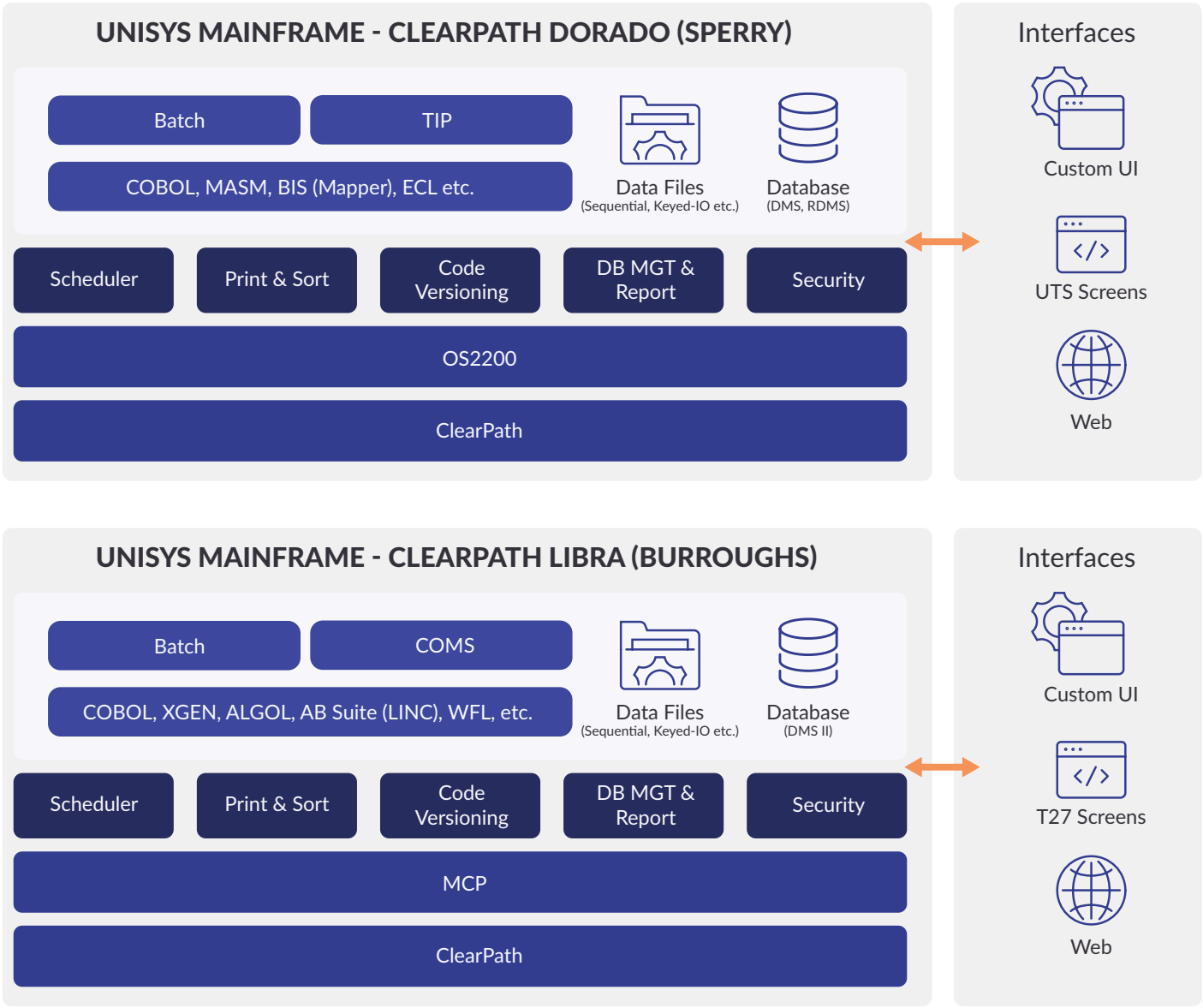
```
$ oc new-app https://github.com/openshift/ruby-hello-world
```

This command creates a deployment of code from the GitHub repository, including everything needed to access the application – service definitions, routes, and deployment. Following the above command, OpenShift will download the source code, compile it, create a new runtime container, deploy the container, and provide access endpoints so users can get to it.

Other interfaces like ODO can make the steps even easier, particularly as code gets more complex, by further abstracting complex Kubernetes and OpenShift concepts for the developer. Thus, this type of interface can be an excellent template for an easy to use/understand interface for Unisys mainframe developers when using OpenShift. It would require implementing one or more operators that this CLI can interact with, which would provide the correct interfaces/mappings between traditional mainframe code and the retrofitted code. This could include methods to generate a "converted" set of code before running it, which would provide necessary feedback to the developer if anything needed to be coded manually.

Mapping Required Unisys Mainframe Features to OpenShift

In practice, given the following components of Unisys OS2200 or MCP systems:



Here is how OpenShift could provide the required features:

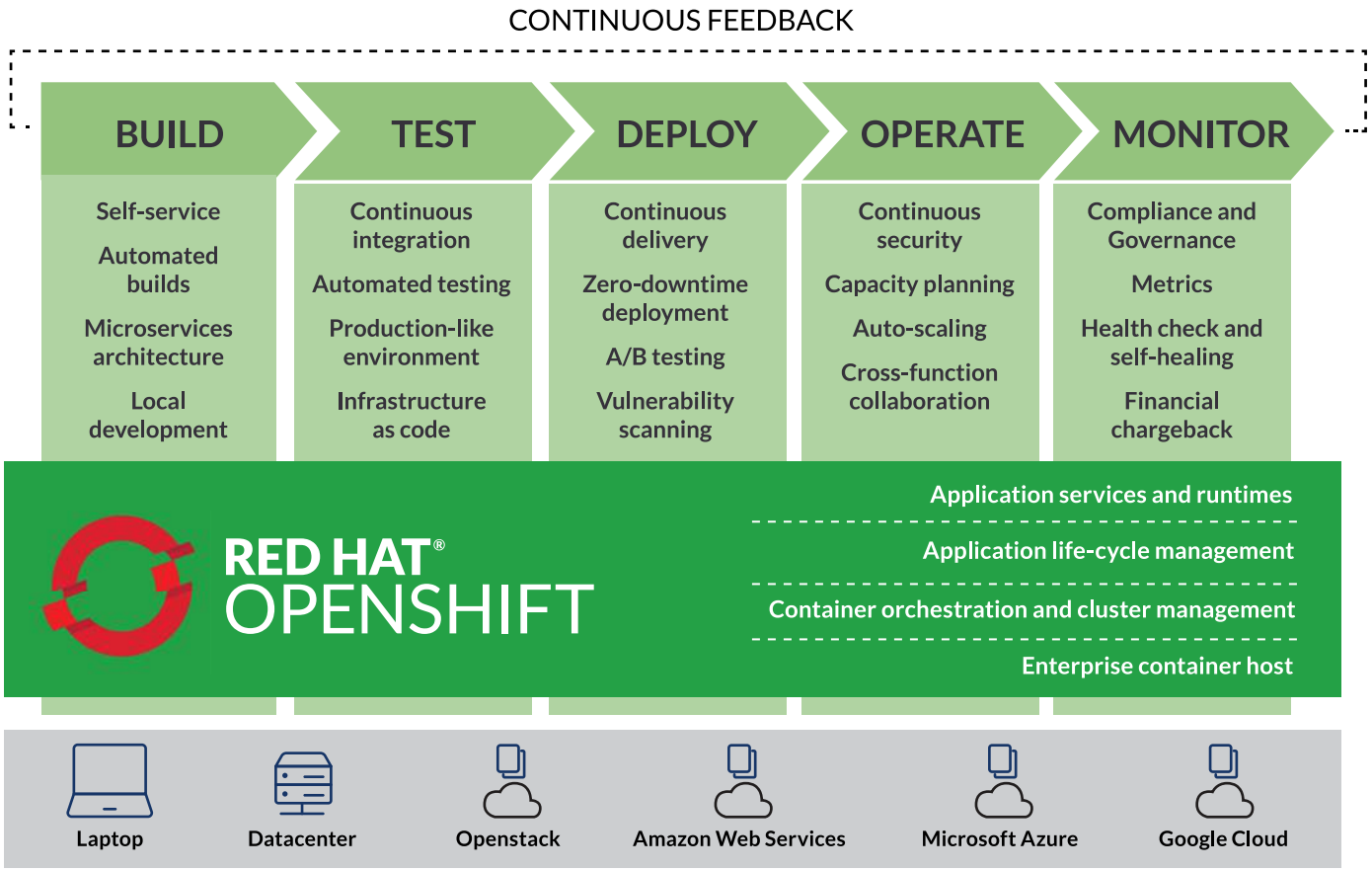
UNISYS TERMINOLOGY	OPENSHIFT EQUIVALENT	BUILT IN?
Batch	OpenShift provides a job interface. This is a scheduled and limited execution of a pod that is expected to end execution within a given time frame.	Yes
TIP	Transaction management is an application-provided feature. Jboss has its XA implementation for handling and managing of distributed transactions. Applications are free to include this feature type, but it is not a built-in feature of the platform.	No
COBOL, MASM, FORTRAN, ALGOL	GNU/Linux's GCC provides a compiler/assembler for the native X86 platform. This compiler has COBOL, Fortran and other extensions. Containers with native code will run so any compiled language will execute. Note there may be differences between this compiler and the features it provides and that of OS2200 — typically compatibility flags can be set to control backwards compatibility to be enabled.	Yes
BIS (MAPPER)	While RHEL has several components with some cross functionality (basic reporting like BIS/MAPPER has been done with AWK for decades) a solution to allow old control files to be reused would require a migration and/or custom programming.	No
ECL	ECL is used to control executions, prepare the system for execution etc. These are all features handled by OpenShift and cannot be overridden. ECL commands would have to be mapped to OpenShift equivalents if possible. This is where an API/Operator would help in creating this initial mapping and ensure any change by the user would be reflected in the implementation.	No
Data Files	A pod can be associated with any number of data sources. Typically, a block device is how this is done, but object and file storage is another way to expose "external" content to a deployment. A PVC can be set to only allow access by one POD or by multiple — if data exchange is to be done through files, this would be the method of doing so. Other objects would be creating basic file integration flows using Red Hat Fuse. (Integration) where files can automatically be picked up, validated, delivered depending on policy rules. This would allow a disconnect of how the data moves from application to application. This could also be part of a pipeline, executing consecutive pieces of code as data is processed/mapped into the final result. Red Hat Integration Suite is an add-on to OpenShift. The format of the files on the file system does not matter to OpenShift — the code defines the expected format and features.	Yes

RDMS/Databases	A database is just another execution/pod. External databases can be used as well as databases running inside of OpenShift. The advantages of running the database in the same namespace as the application are that the DB is now part of the same isolation as the application(s) in the same namespace, it is easy to scale the database depending on need and no external integration points creating/maintaining the external DB is required.	Partial
Scheduler	A Kubernetes term called "Scheduler" is part of the control plane and is responsible for determining where workloads should run. It is policy driven, allowing system administrators to set criteria for which node to pick—from Affinity groups to ensuring certain workloads are only scheduled on nodes with the right capabilities. By definition, users do not tell OpenShift when to run a process. A desired state is set using a deployment (or other advanced deployment features) and the Kubernetes scheduler uses this to start execution of the code. Pipelines and controls will change the desired state making OpenShift create the desired changes.	No
Print/Sort (spooling)	This feature can be included in any container. If a more controlled/audited system is required across multiple processes, a common pod/service can be created that exposes the LPD/Cups port allowing other containers to send standard print jobs via this node.	Yes
Code Versioning	All changes are versioned with OpenShift. Deployments are "remembered" allowing the operator to roll back to a previous version of the deployment in case of a failure. Rollback in this case is going back to the exact same binary as was used before; it is not a recompile. Source code traditionally should be source controlled, and often a server like Gitlab or Google's source repository is added to OpenShift providing this feature natively. Other solutions can be external repositories. OpenShift can pull straight from these repositories and compile the code, start a deployment etc. Alternatively, an external build system can create the container images, and push them to a container registry where OpenShift will see a new version is available and automatically deploy it.	No
DB Management and reports	Using vendor provided databases like CrunchDB, a lot of enterprise features such as reporting and advanced management are provided on top of existing open-source solutions like PostgreSQL (in this case). In other words, this may be as simple as picking the right operator from the operator hub.	Yes
Security	OpenShift implements FIPS and other security standards, and is on the way to become FISMA moderate certified. Security is what containers are all about — while allowing the same hardware to execute multiple different processes, keeping them isolated and unable to see content outside of their own domain. OCP does network isolation, data security on top of the process isolation. Depending on the implementation chosen, more granular options may be available if additional security features are needed. Security in this context also includes audit trails.	

Implementing DevOps with OpenShift

DevOps (or more recently DevSecOps, recognizing the critical role of security at every stage of the development lifecycle) is an approach to culture, automation, and a platform designed to increase the speed, flexibility and reliability with which new features and services are delivered. Modern application platforms based on container technology and microservices are critical to DevOps practices, helping deliver secure and innovative software services at the speed of digital business.

The term DevOps was formed by combining the words "development" and "operations", signifying a cultural shift that bridges the gap between development and operation teams which traditionally functioned in siloes. The following diagram shows the various stages of application development. Despite appearing to flow sequentially, DevOps is an iterative process loop based on continuous feedback, constant collaboration, and iterative improvement throughout the entire lifecycle.



OpenShift offers Unisys application migration, development and operations teams a common platform and set of tools for building, deploying, and managing containerized applications on any infrastructure – on-premises or in public, private, or hybrid clouds. OpenShift features and operators make container orchestration and automation much easier:

STAGES	OPENSHIFT FEATURES
Build	**Self-service:** Developers can quickly and easily create applications on demand using their preferred tools without having to wait for IT operations to set up the deployment environment. At the same time, operations still maintains full control over the entire environment. **Automated builds:** Streamlined and automated application builds allow developers to build containers automatically from application source code and binaries in a secure and repeatable manner. **Microservices:** OpenShift Application Runtimes offers a set of certified and supported microservices runtimes, including Spring Boot, WildFly Swarm, Vert.x, and Node.js on OpenShift for building cloud-native applications in addition to built-in support for service discovery, load balancing, single sign-on, and more. **Local development:** Develop and deploy applications locally using the same tools that are used in test and production.

Test	**Continuous integration (CI):** Built-in support for Jenkins CI server lets developers write, test, and integrate code automatically for every change. **Automated testing:** On-demand deployments allow complex, automated testing scenarios by provisioning and testing applications with all their dependencies whenever needed. **Production-like environment:** OpenShift provides an identical technology stack from local development environment to production, which ensures that all applications are tested and verified on the exact same version of middleware, language runtime, and operating system. **Infrastructure as code:** Every aspect of the application and environment is described in a declarative manner that can be versioned and controlled as code in a version control system.
Operate	**Continuous integration and delivery (CI/CD):** Built-in support for pipelines with integration points to existing tools lets teams automate every step of application delivery while taking advantage of their existing process. **Zero-downtime deployment:** Zero-downtime deployments using rolling updates, blue-green deployments, canary releases, and more allow teams to both remove downtime from deployments and deploy frequently in production during normal working hours. **A/B testing:** Full control over application traffic allows teams to serve users multiple versions of their services simultaneously.
Deploy	**Continuous security:** Proactive security patches are provided for Red Hat-certified containers, which can automatically trigger rebuilding and deploying relevant application containers. **Capacity planning:** OpenShift management technology tracks resource utilization trends to inform capacity and what-if scenario planning. **Auto-scaling:** Scaling of applications running on OpenShift is automated through auto-scaling containers based on application load. **Cross-function collaboration:** Granular access control capabilities allow collaboration between development, quality assurance, security, and operations teams by bringing visibility to production environments while operations teams keep control of the actions performed.
Monitor	**Compliance and governance:** Automatically enforces policy across all containers and environments with support from comprehensive insights and detailed logging. **Metrics:** Container metrics provide full visibility into how applications resource usage changes over time. Health checks and self-healing: Health probes allow automatic identification of application issues, allowing quick repair action. **Financial chargeback:** OpenShift tools collect container capacity and utilization data, and generate financial reports to provide visibility into container usage across teams.

The combination of all these features allows faster innovation by supporting many different aspects and practices pertaining to DevOps. Additionally, automated build and deployments (CI/CD), and build and container metrics within OpenShift provide a rapid flow of information and continuous feedback from the build and deployment process back to the migration/development teams. This lets developers detect and rectify anomalies immediately, which is far more effective than fixing them later in production, when fixes have a more critical impact on cost and service delivery.

Ensuring Project Success

Astadia's Legacy Modernization practice has more than 25 years of experience in migrating legacy applications to Open Systems and cloud platforms. Since most Unisys ClearPath MCP and OS2200 mainframe applications are the mission-critical systems of the enterprise, Astadia goes to great lengths to ensure that a thorough and complete project plan is developed for each legacy modernization project we undertake.

Astadia's methodology recognizes the organizational impact that any project of this nature will have on day-to-day operations, as well as the financial and business implications for organizations in both the short and long term. Return on Investment (ROI) and Total Cost of Ownership (TCO) are carefully calculated during this process, and closely managed throughout the project lifecycle.

Astadia's Unisys Mainframe to OpenShift on IBM Cloud Success Methodology has been refined over the course of 200+ successful legacy migration projects, and has become an industry leading approach for our medium and large-scale Unisys mainframe clients.

The key project phases of our Success Methodology are as follows:

Discover

Catalog and analyze all applications, databases, networks, platforms, and processes in the client's portfolio. Document the interrelationships between applications, and all external integration points in the client's configuration. This is a key input to Application Portfolio Management and Application Rationalization.

Design

Astadia's project team analyzes source code, data structures, end-state requirements, and IBM Cloud components to design and architect the solution. The design includes details such as types and instances of OpenShift on IBM Cloud components, transaction loads, batch requirements, programming language conversions and replacements, integration with external systems, third-party software requirements, and planning for future requirements.

Modernize

Astadia employs an iterative, hybrid process of automated code conversion and human intervention to perform the necessary application changes. The technology behind the automation is Astadia's Rules-Based Transformation Engine. This tool preserves the business logic and rules of the client's legacy applications while removing proprietary code that can only execute in the source environment and not in the IBM Cloud. While a minimal-change, lowest-risk approach is employed, some source code or supporting components may be converted to new languages for technical reasons or to comply with client preferences.

Test

Since most Unisys mainframe-migrations to OpenShift on IBM Cloud are typically an as-is migration of applications, testing can focus primarily on two areas: the components that have been changed or replaced, and application performance. For the most part, "parallel" testing is sufficient; that is, the results of operations in the source environment must produce the same results in the target environment, except where differences are expected due to platform changes. For performance testing, the focus is on ensuring the user experience is as good, or better, than the legacy environment and batch processes completed within acceptable timeframes.

Implement

When migrated applications have been tested, verified, and optimized, the process of deploying those applications may begin. Many deployment activities are initiated in parallel with earlier phases—things like creating and configuring the OpenShift on IBM Cloud component instances, installing and configuring mainframe emulation software, migrating static data, and other infrastructure or framework activities. In some cases, environments may be replicated to achieve this, or existing environments may be re-purposed. The specifics of this may depend upon application and data characteristics and client preferences. After dynamic data is migrated and validated, cutover to Production mode can be completed.

Manage

Astadia offers a full range of managed services solutions. Having gained significant application knowledge from the architecting and implementing of OpenShift on IBM Cloud solutions, Astadia is well-suited to take on the burden of managing and maintaining the migrated applications and their OpenShift on IBM Cloud environments, or the dual environment in the case of a partial migration. This offers clients an opportunity to focus their development efforts on strategic initiatives as well as address concerns of finding programmers skilled in maintaining the legacy components still in use.

Conclusion

Red Hat OpenShift on IBM Cloud Platform and Astadia migration/modernization services combine to deliver a perfect next-generation platform for your Unisys ClearPath MCP and OS2200 mainframe applications portfolio.

Once the mainframe application set has been fully deployed on OpenShift on IBM Cloud Platform, you will have the freedom to re-engineer traditional applications into a more contemporary computing style, modernize legacy interfaces and integrate with other applications. In addition, many new services, like mobile and wireless, can be easily connected to OpenShift on IBM Cloud platform, thus enhancing the overall power of your new cloud computing environment. Your investment will serve to support both the current needs and the future requirements of your business.

There is no need to tackle this alone. Astadia has the proven experience, skilled experts and technology to help you successfully complete Unisys legacy migration projects of all scopes and sizes.

We would be pleased to discuss your specific legacy migration needs and how Astadia can help you leverage OpenShift on the IBM Cloud Platform.

To connect with an Astadia expert, please contact us at:

info@astadia.com

www.astadia.com

(877) 727-8234

Astadia would like to express its deep gratitude for the support and assistance of our business partners Red Hat, Inc. and IBM, Inc. in the creation of this document, including their generosity in allowing us to repurpose portions of their proprietary content and graphic images.

ABOUT ASTADIA

Astadia has been a leader in legacy modernization since 1994 and has successfully completed more than 200 mainframe migration/modernization projects. Our consistent success has allowed us to develop a comprehensive methodology combined with proprietary software tools and techniques – not to mention the extensive hands-on expertise that can only be gained from more than 25 years of migrating mission-critical applications and databases. We're pleased to share some of that experience with you through our Mainframe to Cloud Modernization Series of Reference Architecture Guides, webinars, whitepapers and more.

Visit our website at www.astadia.com for additional information.