Red Hat Openshift Container Platform

­

{{client}}

Assessment Report

{{startmonth}}-{{endmonth}} {{year}}

SPGI CSM





**Confidentiality, Copyright, and Disclaimer**

Copyright IBM, All Rights Reserved. No part of the work covered by the copyright herein may be reproduced or used in any form or by any means – graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems without permission in writing from IBM except as is required to share this information as provided with the confidential parties.

**Trademarks**

Trademarked names may appear throughout this document. Rather than list the names and entities that own the trademarks or insert a trademark symbol with each mention of the trademarked name, the names are used only for editorial purposes and to the benefit of the trademark owner with no intention of infringing upon that trademark.

**Audience**

This document is intended for Client technical staff responsible for the environment.

Table of Contents

[Table of Contents 2](#_Toc1421582786)

[Introduction 4](#_Toc1907547086)

[Purpose of this document 5](#_Toc1095502859)

[Introduction to OpenShift Container Platform 5](#_Toc1037648541)

[About Kubernetes 5](#_Toc1546421142)

[The benefits of containerized applications 6](#_Toc1020448253)

[Operating system benefits 6](#_Toc403508428)

[Deployment and scaling benefits 6](#_Toc61272115)

[OpenShift Container Platform overview 6](#_Toc61068523)

[Custom operating system 7](#_Toc1209813405)

[Simplified installation and update process 7](#_Toc2069506652)

[Other key features 7](#_Toc1569659176)

[OpenShift Container Platform lifecycle 8](#_Toc556827240)

[OpenShift Container Platform installation overview 8](#_Toc1640799609)

[About the OpenShift Update Service 9](#_Toc402134361)

[Understanding the OpenShift Container Platform control plane 10](#_Toc1582432141)

[Machine roles in OpenShift Container Platform 10](#_Toc529572712)

[Cluster workers 11](#_Toc201247444)

[Cluster masters 11](#_Toc1614272527)

[About RHCOS 12](#_Toc984781121)

[Key RHCOS features 13](#_Toc2145396106)

[Understanding Authentication 14](#_Toc1313095420)

[Users 14](#_Toc913250400)

[Groups 14](#_Toc260189231)

[Using RBAC to define and apply permissions 15](#_Toc480156834)

[RBAC overview 15](#_Toc267647671)

[Default cluster roles 16](#_Toc1523110315)

[Understanding networking 17](#_Toc662943897)

[OpenShift Container Platform DNS 17](#_Toc1607357348)

[Cluster Network Operator 18](#_Toc1483170119)

[DNS Operator 18](#_Toc439474651)

[Integrated OpenShift Container Platform registry 18](#_Toc210998684)

[Findings 18](#_Toc1894707640)

[Operational Interview 19](#_Toc2098774550)

[General Scoring 19](#_Toc1318924380)

[Cluster 19](#_Toc395444650)

[Container 22](#_Toc1238698431)

[Development 23](#_Toc1154143437)

[Network 26](#_Toc922602728)

[High Availability – Disaster Recovery 27](#_Toc663765372)

[Red Hat Insights 28](#_Toc1552325170)

[Workloads are still using the deprecated APIs which will be removed in the next release 28](#_Toc612118024)

[IBM Cloud Monitoring 29](#_Toc425454964)

[Cluster Capacity Planning 30](#_Toc2046739106)

[Limit overcommit 30](#_Toc1669586814)

[Use of Allocated Resources 31](#_Toc936308199)

[Unused Requested Resources (CPU / RAM) 31](#_Toc1630608396)

[Container Golden Signals 32](#_Toc672124581)

[Node Resources Usage 32](#_Toc1758384019)

[Network Latency per Pod 33](#_Toc1202314363)

[CPU & Memory used by applications vs request (Monitor applications) 33](#_Toc1692053797)

[Key Recommendations 34](#_Toc1328425474)

[Openshift Architecture 35](#_Toc808360111)

[Openshift Security 35](#_Toc1550608433)

[Resiliency Tests 36](#_Toc327895936)

[Whitespace Deployment – External Secrets 37](#_Toc1311130573)

[IBM Next Step Proposals 38](#_Toc1614152817)

[IBM Workshop and Education enablement sessions 39](#_Toc941750953)

[Application Resource Management - Turbonomics 40](#_Toc1081967074)

[Modernization in Openshift of Legacy Applications in WebSphere 41](#_Toc97576300)

[Cloud Pak for Multicloud Management – Red Hat Advanced Cluster Management for Kubernetes. 42](#_Toc1937682427)

[Modernization of IBM products into Openshift 43](#_Toc960542260)

# Introduction

## Purpose of this document

The OpenShift Container Platform Assessment (OCPA) is a specific collaboration delivered by IBM experts designed to guarantee the efficacy of your Container Platform strategy by evaluating all the relevant aspects in the implementation, including technical, organizational, and operational controls. Through on-site, in-person interviews and technical examination, OCPA results in the creation of a results report that is customized to your business. The information obtained accounts for your organization’s resources status in the infrastructure setup, operational excellence, performance metrics, reliability, high availability, and security design wherever possible. All this information captures opportunities for improvement to contribute to best practices adoption and ensure the success of technology usage.

This document contains all the detailed information and results obtained during the different phases of the assessment specifically for your OCP environment. In the next sections you will find all the recommendations, improvements points and topics that have been analyzed, including all the information with the status and checks results even when the area reviewed is in healthy state. This document complements the OCP results summary deck, which is focused in the key findings and improvement points summary.

## Introduction to OpenShift Container Platform[[1]](#footnote-1)

OpenShift Container Platform is a platform for developing and running containerized applications. It is designed to allow applications and the data centers that support them to expand from just a few machines and applications to thousands of machines that serve millions of clients.

With its foundation in Kubernetes, OpenShift Container Platform incorporates the same technology that serves as the engine for massive telecommunications, streaming video, gaming, banking, and other applications. Its implementation in open Red Hat technologies lets you extend your containerized applications beyond a single cloud to on-premises and multi-cloud environments.

### About Kubernetes

Although container images and the containers that run from them are the primary building blocks for modern application development, to run them at scale requires a reliable and flexible distribution system. Kubernetes is the defacto standard for orchestrating containers.

Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications. The general concept of Kubernetes is fairly simple:

* Start with one or more worker nodes to run the container workloads.
* Manage the deployment of those workloads from one or more master nodes.
* Wrap containers in a deployment unit called a pod. Using pods provides extra metadata with the container and offers the ability to group several containers in a single deployment entity.
* Create special kinds of assets. For example, services are represented by a set of pods and a policy that defines how they are accessed. This policy allows containers to connect to the services that they need even if they do not have the specific IP addresses for the services. Replication controllers are another special asset that indicates how many pod replicas are required to run at a time. You can use this capability to automatically scale your application to adapt to its current demand.

In only a few years, Kubernetes has seen massive cloud and on-premise adoption. The open source development model allows many people to extend Kubernetes by implementing different technologies for components such as networking, storage, and authentication.

### The benefits of containerized applications

Using containerized applications offers many advantages over using traditional deployment methods. Where applications were once expected to be installed on operating systems that included all their dependencies, containers let an application carry their dependencies with them. Creating containerized applications offers many benefits.

#### Operating system benefits

Containers use small, dedicated Linux operating systems without a kernel. Their file system, networking, cgroups, process tables, and namespaces are separate from the host Linux system, although they can integrate with the host seamlessly when necessary. Being based on Linux, allows containers to use all the advantages that come with the open source development model of rapid innovation.

Because each container uses a dedicated operating system, you can deploy applications that require conflicting software dependencies on the same host. Each container carries its own dependent software and manages its own interfaces, such as networking and file systems, so applications never need to compete for those assets.

#### Deployment and scaling benefits

If you employ rolling upgrades between major releases of your application, you can continuously improve your applications without downtime and still maintain compatibility with the current release.

You can also deploy and test a new version of an application alongside the existing version. If the container passes your tests, simply deploy more new containers and remove the old ones.

Since all the software dependencies for an application are resolved within the container itself, you can use a standardized operating system on each host in your data center. You do not need to configure a specific operating system for each application host. When your data center needs more capacity, you can deploy another generic host system.

Similarly, scaling containerized applications is simple. OpenShift Container Platform offers a simple, standard way of scaling any containerized service. For example, if you build applications as a set of microservices rather than large, monolithic applications, you can scale the individual microservices individually to meet demand. This capability allows you to scale only the required services instead of the entire application, which can allow you to meet application demands while using minimal resources.

### OpenShift Container Platform overview

OpenShift Container Platform provides enterprise-ready enhancements to Kubernetes, including the following enhancements:

* Hybrid cloud deployments. You can deploy OpenShift Container Platform clusters to a variety of public cloud platforms or in your data center.
* Integrated Red Hat technology. Major components in OpenShift Container Platform come from Red Hat Enterprise Linux (RHEL) and related Red Hat technologies. OpenShift Container Platform benefits from the intense testing and certification initiatives for Red Hat’s enterprise quality software.
* Open source development model. Development is completed in the open, and the source code is available from public software repositories. This open collaboration fosters rapid innovation and development.

Although Kubernetes excels at managing your applications, it does not specify or manage platform-level requirements or deployment processes. Powerful and flexible platform management tools and processes are important benefits that OpenShift Container Platform 4.8 offers. The following sections describe some unique features and benefits of OpenShift Container Platform.

#### Custom operating system

OpenShift Container Platform uses Red Hat Enterprise Linux CoreOS (RHCOS), a container-oriented operating system that combines some of the best features and functions of the CoreOS and Red Hat Atomic Host operating systems. RHCOS is specifically designed for running containerized applications from OpenShift Container Platform and works with new tools to provide fast installation, Operator-based management, and simplified upgrades.

RHCOS includes:

* Ignition, which is used by OpenShift Container Platform as a firstboot system configuration for initially bringing up and configuring machines.
* CRI-O, a Kubernetes native container runtime implementation that integrates closely with the operating system to deliver an efficient and optimized Kubernetes experience. CRI-O provides facilities for running, stopping, and restarting containers. It fully replaces the Docker Container Engine, which was used in OpenShift Container Platform 3.
* Kubelet, the primary node agent for Kubernetes, responsible for launching and monitoring containers.
* Atomic Upgrades managed

In OpenShift Container Platform 4.8, it is mandatory to use RHCOS for all control plane machines, but you can use Red Hat Enterprise Linux (RHEL) as the operating system for compute machines, which are also known as worker machines. If you choose to use RHEL for the worker machines, you must perform manual system maintenance because RHEL upgrades will not be managed by the cluster.

#### Simplified installation and update process

With OpenShift Container Platform 4.8, if you have an account with the right permissions, you can deploy a production cluster in supported clouds by running a single command and providing a few values. You can also customize your cloud installation or install your cluster in your data center if you use a supported platform.

For clusters that use RHCOS for all machines, updating, or upgrading, OpenShift Container Platform is a simple, highly-automated process. Because OpenShift Container Platform completely controls the systems and services that run on each machine, including the operating system itself, from a central control plane, upgrades are designed to become automatic events. If your cluster contains RHEL worker machines, the control plane benefits from the streamlined update process, but you must perform more tasks to upgrade the RHEL machines.

#### Other key features

Operators are both the fundamental unit of the OpenShift Container Platform 4.8 code base and a convenient way to deploy applications and software components for your applications to use. In OpenShift Container Platform, Operators serve as the platform foundation and remove the need for manual upgrades of operating systems and control plane applications. OpenShift Container Platform Operators such as the Cluster Version Operator and Machine Config Operator allow simplified, cluster-wide management of those critical components.

Operator Lifecycle Manager (OLM) and the OperatorHub provide facilities for storing and distributing Operators to people developing and deploying applications.

The Red Hat Quay Container Registry is a Quay.io container registry that serves most of the container images and Operators to OpenShift Container Platform clusters. Quay.io is a public registry version of Red Hat Quay that stores millions of images and tags.

Other enhancements to Kubernetes in OpenShift Container Platform include improvements in software defined networking (SDN), authentication, log aggregation, monitoring, and routing. OpenShift Container Platform also offers a comprehensive web console and the custom OpenShift CLI (oc) interface.

#### OpenShift Container Platform lifecycle

The following figure illustrates the basic OpenShift Container Platform lifecycle:

* Creating an OpenShift Container Platform cluster
* Managing the cluster
* Developing and deploying applications
* Scaling up applications

Diagram

Description automatically generated with low confidence

## OpenShift Container Platform installation overview

The OpenShift Container Platform installation program offers you flexibility. You can use the installation program to deploy a cluster on infrastructure that the installation program provisions and the cluster maintains or deploy a cluster on infrastructure that you prepare and maintain.

These two basic types of OpenShift Container Platform clusters are frequently called installer-provisioned infrastructure clusters and user-provisioned infrastructure clusters.

Both types of clusters have the following characteristics:

* Highly available infrastructure with no single points of failure is available by default
* Administrators maintain control over what updates are applied and when

You use the same installation program to deploy both types of clusters. The main assets generated by the installation program are the Ignition config files for the bootstrap, master, and worker machines. With these three configurations and correctly configured infrastructure, you can start an OpenShift Container Platform cluster.

The OpenShift Container Platform installation program uses a set of targets and dependencies to manage cluster installation. The installation program has a set of targets that it must achieve, and each target has a set of dependencies. Because each target is only concerned with its own dependencies, the installation program can act to achieve multiple targets in parallel. The ultimate target is a running cluster. By meeting dependencies instead of running commands, the installation program is able to recognize and use existing components instead of running the commands to create them again.

The following diagram shows a subset of the installation targets and dependencies:

Graphical user interface, application, Teams

Description automatically generated

After installation, each cluster machine uses Red Hat Enterprise Linux CoreOS (RHCOS) as the operating system. RHCOS is the immutable container host version of Red Hat Enterprise Linux (RHEL) and features a RHEL kernel with SELinux enabled by default. It includes the kubelet, which is the Kubernetes node agent, and the CRI-O container runtime, which is optimized for Kubernetes.

Every control plane machine in an OpenShift Container Platform 4.8 cluster must use RHCOS, which includes a critical first-boot provisioning tool called Ignition. This tool enables the cluster to configure the machines. Operating system updates are delivered as an Atomic OSTree repository that is embedded in a container image that is rolled out across the cluster by an Operator. Actual operating system changes are made in-place on each machine as an atomic operation by using rpm-ostree. Together, these technologies enable OpenShift Container Platform to manage the operating system like it manages any other application on the cluster, via in-place upgrades that keep the entire platform up-to-date. These in-place updates can reduce the burden on operations teams.

If you use RHCOS as the operating system for all cluster machines, the cluster manages all aspects of its components and machines, including the operating system. Because of this, only the installation program and the Machine Config Operator can change machines. The installation program uses Ignition config files to set the exact state of each machine, and the Machine Config Operator completes more changes to the machines, such as the application of new certificates or keys, after installation.

## About the OpenShift Update Service

The OpenShift Update Service (OSUS) provides over-the-air updates to OpenShift Container Platform, including Red Hat Enterprise Linux CoreOS (RHCOS). It provides a graph, or diagram, that contains the vertices of component Operators and the edges that connect them. The edges in the graph show which versions you can safely update to. The vertices are update payloads that specify the intended state of the managed cluster components.

The Cluster Version Operator (CVO) in your cluster checks with the OpenShift Update Service to see the valid updates and update paths based on current component versions and information in the graph. When you request an update, the CVO uses the release image for that update to upgrade your cluster. The release artifacts are hosted in Quay as container images.

To allow the OpenShift Update Service to provide only compatible updates, a release verification pipeline drives automation. Each release artifact is verified for compatibility with supported cloud platforms and system architectures, as well as other component packages. After the pipeline confirms the suitability of a release, the OpenShift Update Service notifies you that it is available.

Because the update service displays all valid updates, you must not force an update to a version that the update service does not display.

Two controllers run during continuous update mode. The first controller continuously updates the payload manifests, applies the manifests to the cluster, and outputs the controlled rollout status of the Operators to indicate whether they are available, upgrading, or failed. The second controller polls the OpenShift Update Service to determine if updates are available.

Reverting your cluster to a previous version, or a rollback, is not supported. Only upgrading to a newer version is supported.

During the upgrade process, the Machine Config Operator (MCO) applies the new configuration to your cluster machines. The MCO cordons the number of nodes as specified by the maxUnavailable field on the machine configuration pool and marks them as unavailable. By default, this value is set to 1. The MCO then applies the new configuration and reboots the machine.

If you use Red Hat Enterprise Linux (RHEL) machines as workers, the MCO does not update the kubelet because you must update the OpenShift API on the machines first.

With the specification for the new version applied to the old kubelet, the RHEL machine cannot return to the Ready state. You cannot complete the update until the machines are available. However, the maximum number of unavailable nodes is set to ensure that normal cluster operations can continue with that number of machines out of service.

The OpenShift Update Service is composed of an Operator and one or more application instances.

## Understanding the OpenShift Container Platform control plane

The control plane, which is composed of master machines, manages the OpenShift Container Platform cluster. The control plane machines manage workloads on the compute machines, which are also known as worker machines. The cluster itself manages all upgrades to the machines by the actions of the Cluster Version Operator, the Machine Config Operator, and a set of individual Operators.

### Machine roles in OpenShift Container Platform

OpenShift Container Platform assigns hosts different roles. These roles define the function of the machine within the cluster. The cluster contains definitions for the standard master and worker role types.

#### Cluster workers

In a Kubernetes cluster, the worker nodes are where the actual workloads requested by Kubernetes users run and are managed. The worker nodes advertise their capacity and the scheduler, which is part of the master services, determines on which nodes to start containers and pods. Important services run on each worker node, including CRI-O, which is the container engine, Kubelet, which is the service that accepts and fulfills requests for running and stopping container workloads, and a service proxy, which manages communication for pods across workers.

In OpenShift Container Platform, machine sets control the worker machines. Machines with the worker role drive compute workloads that are governed by a specific machine pool that autoscales them. Because OpenShift Container Platform has the capacity to support multiple machine types, the worker machines are classed as compute machines. In this release, the terms worker machine and compute machine are used interchangeably because the only default type of compute machine is the worker machine. In future versions of OpenShift Container Platform, different types of compute machines, such as infrastructure machines, might be used by default.

#### Cluster masters

In a Kubernetes cluster, the master nodes run services that are required to control the Kubernetes cluster. In OpenShift Container Platform, the master machines are the control plane. They contain more than just the Kubernetes services for managing the OpenShift Container Platform cluster. Because all of the machines with the control plane role are master machines, the terms master and control plane are used interchangeably to describe them. Instead of being grouped into a machine set, master machines are defined by a series of standalone machine API resources. Extra controls apply to master machines to prevent you from deleting all master machines and breaking your cluster.

Exactly three master nodes must be used for all production deployments.

Services that fall under the Kubernetes category on the master include the Kubernetes API server, etcd, the Kubernetes controller manager, and the Kubernetes scheduler.

|  |  |
| --- | --- |
| Component | Description |
| Kubernetes API server | The Kubernetes API server validates and configures the data for pods, services, and replication controllers. It also provides a focal point for the shared state of the cluster. |
| etcd | etcd stores the persistent master state while other components watch etcd for changes to bring themselves into the specified state. |
| Kubernetes controller manager | The Kubernetes controller manager watches etcd for changes to objects such as replication, namespace, and service account controller objects, and then uses the API to enforce the specified state. Several such processes create a cluster with one active leader at a time. |
| Kubernetes scheduler | The Kubernetes scheduler watches for newly created pods without an assigned node and selects the best node to host the pod. |

Some of these services on the master machines run as systemd services, while others run as static pods.

Systemd services are appropriate for services that you need to always come up on that particular system shortly after it starts. For master machines, those include sshd, which allows remote login. It also includes services such as:

The CRI-O container engine (crio), which runs and manages the containers. OpenShift Container Platform 4.8 uses CRI-O instead of the Docker Container Engine.

Kubelet (kubelet), which accepts requests for managing containers on the machine from master services.

CRI-O and Kubelet must run directly on the host as systemd services because they need to be running before you can run other containers.

## About RHCOS

Red Hat Enterprise Linux CoreOS (RHCOS) represents the next generation of single-purpose container operating system technology. Created by the same development teams that created Red Hat Enterprise Linux Atomic Host and CoreOS Container Linux, RHCOS combines the quality standards of Red Hat Enterprise Linux (RHEL) with the automated, remote upgrade features from Container Linux.

RHCOS is supported only as a component of OpenShift Container Platform 4.8 for all OpenShift Container Platform machines. RHCOS is the only supported operating system for OpenShift Container Platform control plane, or master, machines. While RHCOS is the default operating system for all cluster machines, you can create compute machines, which are also known as worker machines, that use RHEL as their operating system. There are two general ways RHCOS is deployed in OpenShift Container Platform 4.8:

* If you install your cluster on infrastructure that the cluster provisions, RHCOS images are downloaded to the target platform during installation, and suitable Ignition config files, which control the RHCOS configuration, are used to deploy the machines.
* If you install your cluster on infrastructure that you manage, you must follow the installation documentation to obtain the RHCOS images, generate Ignition config files, and use the Ignition config files to provision your machines.

### Key RHCOS features

The following list describes key features of the RHCOS operating system:

* **Based on RHEL:** The underlying operating system consists primarily of RHEL components. The same quality, security, and control measures that support RHEL also support RHCOS. For example, RHCOS software is in RPM packages, and each RHCOS system starts up with a RHEL kernel and a set of services that are managed by the systemd init system.
* **Controlled immutability:** Although it contains RHEL components, RHCOS is designed to be managed more tightly than a default RHEL installation. Management is performed remotely from the OpenShift Container Platform cluster. When you set up your RHCOS machines, you can modify only a few system settings. This controlled immutability allows OpenShift Container Platform to store the latest state of RHCOS systems in the cluster so it is always able to create additional machines and perform updates based on the latest RHCOS configurations.
* **CRI-O container runtime:** Although RHCOS contains features for running the OCI- and libcontainer-formatted containers that Docker requires, it incorporates the CRI-O container engine instead of the Docker container engine. By focusing on features needed by Kubernetes platforms, such as OpenShift Container Platform, CRI-O can offer specific compatibility with different Kubernetes versions. CRI-O also offers a smaller footprint and reduced attack surface than is possible with container engines that offer a larger feature set. At the moment, CRI-O is the only engine available within OpenShift Container Platform clusters.
* **Set of container tools:** For tasks such as building, copying, and otherwise managing containers, RHCOS replaces the Docker CLI tool with a compatible set of container tools. The podman CLI tool supports many container runtime features, such as running, starting, stopping, listing, and removing containers and container images. The skopeo CLI tool can copy, authenticate, and sign images. You can use the crictl CLI tool to work with containers and pods from the CRI-O container engine. While direct use of these tools in RHCOS is discouraged, you can use them for debugging purposes.
* **rpm-ostree upgrades:** RHCOS features transactional upgrades using the rpm-ostree system. Updates are delivered by means of container images and are part of the OpenShift Container Platform update process. When deployed, the container image is pulled, extracted, and written to disk, then the bootloader is modified to boot into the new version. The machine will reboot into the update in a rolling manner to ensure cluster capacity is minimally impacted.
* **bootupd firmware and bootloader updater:** Package managers and hybrid systems such as rpm-ostree do not update the firmware or the bootloader. With bootupd, RHCOS users have access to a cross-distribution, system-agnostic update tool that manages firmware and boot updates in UEFI and legacy BIOS boot modes that run on modern architectures, such as x86\_64, ppc64le, and aarch64.
* **Updated through the Machine Config Operator:** In OpenShift Container Platform, the Machine Config Operator handles operating system upgrades. Instead of upgrading individual packages, as is done with yum upgrades, rpm-ostree delivers upgrades of the OS as an atomic unit. The new OS deployment is staged during upgrades and goes into effect on the next reboot. If something goes wrong with the upgrade, a single rollback and reboot returns the system to the previous state. RHCOS upgrades in OpenShift Container Platform are performed during cluster updates.

For RHCOS systems, the layout of the rpm-ostree file system has the following characteristics:

* /usr is where the operating system binaries and libraries are stored and is read-only. We do not support altering this.
* /etc, /boot, /var are writable on the system but only intended to be altered by the Machine Config Operator.
* /var/lib/containers is the graph storage location for storing container images.

## Understanding Authentication

For users to interact with OpenShift Container Platform, they must first authenticate to the cluster. The authentication layer identifies the user associated with requests to the OpenShift Container Platform API. The authorization layer then uses information about the requesting user to determine if the request is allowed.

As an administrator, you can configure authentication for OpenShift Container Platform.

### Users

A user in OpenShift Container Platform is an entity that can make requests to the OpenShift Container Platform API. An OpenShift Container Platform User object represents an actor which can be granted permissions in the system by adding roles to them or to their groups. Typically, this represents the account of a developer or administrator that is interacting with OpenShift Container Platform.

Several types of users can exist:

|  |  |
| --- | --- |
| Component | Description |
| Regular Users | This is the way most interactive OpenShift Container Platform users are represented. Regular users are created automatically in the system upon first login or can be created via the API. Regular users are represented with the User object. Examples: joe alice |
| System Users | Many of these are created automatically when the infrastructure is defined, mainly for the purpose of enabling the infrastructure to interact with the API securely. They include a cluster administrator (with access to everything), a per-node user, users for use by routers and registries, and various others. Finally, there is an anonymous system user that is used by default for unauthenticated requests. Examples: system:admin system:openshift-registry system:node:node1.example.com |
| Service Accounts | These are special system users associated with projects; some are created automatically when the project is first created, while project administrators can create more for the purpose of defining access to the contents of each project. Service accounts are represented with the ServiceAccount object. Examples: system:serviceaccount:default:deployer system:serviceaccount:foo:builder |

Each user must authenticate in some way to access OpenShift Container Platform. API requests with no authentication or invalid authentication are authenticated as requests by the anonymous system user. Once authenticated, policy determines what the user is authorized to do.

### Groups

A user can be assigned to one or more groups, each of which represent a certain set of users. Groups are useful when managing authorization policies to grant permissions to multiple users at once, for example allowing access to objects within a project, versus granting them to users individually.

In addition to explicitly defined groups, there are also system groups, or virtual groups, that are automatically provisioned by the cluster.

The following default virtual groups are most important:

|  |  |
| --- | --- |
| Virtual Group | Description |
| system:authenticated | Automatically associated with all authenticated users. |
| system:authenticated:oauth | Automatically associated with all users authenticated with an OAuth access token. |
| Service Accounts | Automatically associated with all unauthenticated users. |

## Using RBAC to define and apply permissions

### RBAC overview

Role-based access control (RBAC) objects determine whether a user is allowed to perform a given action within a project.

Cluster administrators can use the cluster roles and bindings to control who has various access levels to the OpenShift Container Platform platform itself and all projects.

Developers can use local roles and bindings to control who has access to their projects. Note that authorization is a separate step from authentication, which is more about determining the identity of who is taking the action.

Authorization is managed using:

|  |  |
| --- | --- |
| Authorization Object | Description |
| Rules | Sets of permitted verbs on a set of objects. For example, whether a user or service account can create pods. |
| Roles | Collections of rules. You can associate, or bind, users and groups to multiple roles. |
| Bindings | Associations between users and/or groups with a role. |

There are two levels of RBAC roles and bindings that control authorization:

|  |  |
| --- | --- |
| RBAC Level | Description |
| Cluster RBAC | Roles and bindings that are applicable across all projects. Cluster roles exist cluster-wide, and cluster role bindings can reference only cluster roles. |
| Local RBAC | Roles and bindings that are scoped to a given project. While local roles exist only in a single project, local role bindings can reference both cluster and local roles. |

A cluster role binding is a binding that exists at the cluster level. A role binding exists at the project level. The cluster role view must be bound to a user using a local role binding for that user to view the project. Create local roles only if a cluster role does not provide the set of permissions needed for a particular situation.

This two-level hierarchy allows reuse across multiple projects through the cluster roles while allowing customization inside of individual projects through local roles.

During evaluation, both the cluster role bindings and the local role bindings are used. For example:

* Cluster-wide "allow" rules are checked.
* Locally-bound "allow" rules are checked.
* Deny by default.

### Default cluster roles

OpenShift Container Platform includes a set of default cluster roles that you can bind to users and groups cluster-wide or locally. You can manually modify the default cluster roles, if required.

|  |  |
| --- | --- |
| Default cluster role | Description |
| admin | A project manager. If used in a local binding, an admin has rights to view any resource in the project and modify any resource in the project except for quota. |
| basic-user | A user that can get basic information about projects and users. |
| cluster-admin | A super-user that can perform any action in any project. When bound to a user with a local binding, they have full control over quota and every action on every resource in the project. |
| cluster-status | A user that can get basic cluster status information. |
| edit | A user that can modify most objects in a project but does not have the power to view or modify roles or bindings. |
| self-provisioner | A user that can create their own projects. |
| view | A user who cannot make any modifications, but can see most objects in a project. They cannot view or modify roles or bindings. |

Be mindful of the difference between local and cluster bindings. For example, if you bind the cluster-admin role to a user by using a local role binding, it might appear that this user has the privileges of a cluster administrator. This is not the case. Binding the cluster-admin to a user in a project grants super administrator privileges for only that project to the user. That user has the permissions of the cluster role admin, plus a few additional permissions like the ability to edit rate limits, for that project. This binding can be confusing via the web console UI, which does not list cluster role bindings that are bound to true cluster administrators. However, it does list local role bindings that you can use to locally bind cluster-admin.

The relationships between cluster roles, local roles, cluster role bindings, local role bindings, users, groups and service accounts are illustrated below.

Timeline

Description automatically generated

## Understanding networking

Kubernetes ensures that pods are able to network with each other and allocates each pod an IP address from an internal network. This ensures all containers within the pod behave as if they were on the same host. Giving each pod its own IP address means that pods can be treated like physical hosts or virtual machines in terms of port allocation, networking, naming, service discovery, load balancing, application configuration, and migration.

### OpenShift Container Platform DNS

If you are running multiple services, such as front-end and back-end services for use with multiple pods, environment variables are created for usernames, service IPs, and more so the front-end pods can communicate with the back-end services. If the service is deleted and recreated, a new IP address can be assigned to the service and requires the front-end pods to be recreated to pick up the updated values for the service IP environment variable. Additionally, the back-end service must be created before any of the front-end pods to ensure that the service IP is generated properly, and that it can be provided to the front-end pods as an environment variable.

For this reason, OpenShift Container Platform has a built-in DNS so that the services can be reached by the service DNS as well as the service IP/port.

### Cluster Network Operator

The Cluster Network Operator (CNO) deploys and manages the cluster network components on an OpenShift Container Platform cluster. The CNO implements the network API from the operator.openshift.io API group. The Operator deploys the OpenShift SDN default Container Network Interface (CNI) network provider plug-in, or the default network provider plug-in that you selected during cluster installation, by using a daemon set.

### DNS Operator

The DNS Operator deploys and manages CoreDNS to provide a name resolution service to pods, enabling DNS-based Kubernetes Service discovery in OpenShift. The DNS Operator implements the dns API from the operator.openshift.io API group. The Operator deploys CoreDNS using a daemon set, creates a service for the daemon set, and configures the kubelet to instruct pods to use the CoreDNS service IP address for name resolution.

## Integrated OpenShift Container Platform registry

OpenShift Container Platform provides a built-in container image registry that runs as a standard workload on the cluster. The registry is configured and managed by an infrastructure Operator. It provides an out-of-the-box solution for users to manage the images that run their workloads and runs on top of the existing cluster infrastructure. This registry can be scaled up or down like any other cluster workload and does not require specific infrastructure provisioning. In addition, it is integrated into the cluster user authentication and authorization system, which means that access to create and retrieve images is controlled by defining user permissions on the image resources.

The registry is typically used as a publication target for images built on the cluster, as well as being a source of images for workloads running on the cluster. When a new image is pushed to the registry, the cluster is notified of the new image and other components can react to and consume the updated image.

Image data is stored in two locations. The actual image data is stored in a configurable storage location, such as cloud storage or a filesystem volume. The image metadata, which is exposed by the standard cluster APIs and is used to perform access control, is stored as standard API resources, specifically images and image streams.

# Findings

As the result of the strong collaboration with key representatives of Mutua’s IT infrastructure and Security departments, as well as the evaluation of telemetry provided to IBM Cloud Monitoring and Red Hat Insights, the following key findings are identified.

## Operational Interview

### General Scoring

Chart, bar chart

Description automatically generated

{{image}}

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Total Score** | **Max Achievable Score** | **Compliance** |
| **Cluster** | 120 | 160 | 75% |
| **Container** | 40 | 50 | 40% |
| **Development** | 60 | 120 | 83% |
| **HA-DR** | 10 | 80 | 75% |
| **Network** | 50 | 50 | 40% |
| **Grand Total** | 280 | 460 |  |

The architecture deployed as well as the development and operational processes that Mutua follows and implements, in most of the scenarios evaluated, comply to the available best practices. Nevertheless, in a few cases, recommendations can be applied to enhance the security, functionality, stability, and high availability of the platform.

### Cluster

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| {%tr for Item in table\_cat1 %} | | | | | |
| {{Item.Category}} | | {{Item.Question}} | {{Item.Answer}} | {{Item.Comment}} |  |
| {%tr endfor %} | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| {%tr for Item in table\_cat2 %} | | | | | |
| {{Item.Category}} | | {{Item.Question}} | {{Item.Answer}} | {{Item.Comment}} |  |
| {%tr endfor %} | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| {%tr for Item in table\_cat3 %} | | | | | |
| {{Item.Category}} | | {{Item.Question}} | {{Item.Answer}} | {{Item.Comment}} |  |
| {%tr endfor %} | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| {%tr for Item in table\_cat4 %} | | | | | |
| {{Item.Category}} | | {{Item.Question}} | {{Item.Answer}} | {{Item.Comment}} |  |
| {%tr endfor %} | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| {%tr for Item in table\_cat5 %} | | | | | |
| {{Item.Category}} | | {{Item.Question}} | {{Item.Answer}} | {{Item.Comment}} |  |
| {%tr endfor %} | | | | | |

### Container

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| Container | Do you scan the container image against vulnerabilities? | Yes | PrismaCloud is the selected tool to do it, but they are working on it, not yet on production. | N/A |
| Container | Do you allow deploying containers only from known registries? | Yes | Mutua only uses Quay deployed on OCP. | N/A |
| Container | Do you integrate Runtime Security for your pods? If yes, which tool is used? | Yes | Again PrismaCloud is the selected tool to do it, but not yet on production. | N/A |
| Container | Do you use OCP internal or external container image registries? Or both? | External | Quay is the image registry, deployed on the Management cluster. They don't use the internal OCP image registry | Verify HA for the Image registry. |
| Container | Do you prefer distroless images? | No | Mutua uses RH officials base images. | ["Distroless" images contain only your application and its runtime dependencies. They do not contain package managers, shells or any other programs you would expect to find in a standard Linux distribution. Restricting what's in your runtime container to precisely what's necessary for your app improves the signal to noise of scanners (e.g. CVE) and reduces the burden of establishing provenance to just what you need.](https://github.com/GoogleContainerTools/distroless) |

### Development

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| Development | Do you develop applications’ source code internally? How do you maintain and control it? | Yes | Development is outsourced but all development contractor use the infrastructure and processes defined by Mutua. Jenkins + Bitbucket |  |
| Development | Is the code-built process automated? | Yes | Jenkins + bitbucket (Angel Pablo). No tekton pipelines, no ACM deployment features used for deployment. | Each time a base image is updated, you should also update any downstream container images. The same applies when your code version is updated. Integrate this build process into validation and deployment pipelines such as Jenkins or S2I. |
| Development | Is there the ability to externalize configuration (outside source code)? | No | Mutua does all the configuration for their applications internally. | It is recommended to externalize the configuration, so the deployments don't have dependencies with the runtime. Using configmaps and Secrets is highly recommended, moreover, using an external system to manage applications configuration is a good practice. Refer to the section Whitespace Deployments - External Secrets, for more information about externalize Openshift configmaps and secrets. |
| Development | Do you implement a Liveness probe on your application’s containers? | Yes | N/A | N/A |
| Development | Do you implement a Startup probe on your application’s containers? | No | Mutua does not control when the application has started up, Readiness probe provide that information. | Configure Liveness, Readiness and Startup Probes For better management is recommended to implement startup probes. |
| Development | Do you implement a Readiness probe on your application’s containers? | Yes | N/A | N/A |
| Development | Have you considered implementing prestop hooks? | No | They don't use prestop hooks for graceful shutdown of the applications. They have a JVM OOM debug system that they think could do someting similar. | [Container Lifecycle Hooks  If your application doesn’t gracefully shut down when receiving a SIGTERM you can use this hook to trigger a graceful shutdown. Most programs gracefully shut down when receiving a SIGTERM, but if you are using third-party code or are managing a system you don’t have control over, the preStop hook is a great way to trigger a graceful shutdown without modifying the application.](https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/) |
| Development | Do you run more than one replica per Deployment / Deployment Config? | Yes | Applications have the ability to run more than one replica, most of the applications are stateless. | N/A |
| Development | Do you apply tags to all resources? | Yes | Nodes and Namespaces. | Ensure that your components are tagged, it could be business, security or technical tags and these tags will help to assess or apply relevant policies. |
| Development | Do you implement autoscaling of your critical applications on expected increased load (e.g. End of month)? | No | Autoscale is not configured, every scale up/down is manual | Automatic scale could be an improvement for the future. |
| Development | Do you specify the security context of your pod/container? | No | Mutua doesn't define specific SCC, use the default OCP SCC. | [SC for a Pod/Container](https://kubernetes.io/docs/tasks/configure-pod-container/security-context/) |
| Development | Do you use the default namespace for deploying applications? | No | No deployments in the default namespace. | N/A |

### Network

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| Network | Applications in a containerized environment should not be heavily reliant on specific IP/port bindings. Do you foresee or face any limitation in the specific rule of thumb for your applications? | No | Mutua doesn't have any hardcoded IPs/ports for the deployments. | N/A |
| Network | Do you distribute ingress traffic? | Yes | Yes, infrastructure nodes has the routers per environment. | To distribute HTTP or HTTPS traffic to your applications, use multiple ingress controllers.   https://rubix.nl/setting-up-multiple-ingress-controllers-on-openshift-4-x/ |
| Network | Do you secure your exposed applications with a web application firewall (WAF)? | Yes | Imperva is the WAF Mutua is using. | N/A |
| Network | Do you control traffic flow with network policies? | Yes | Mutua doesn't allow connections between namespaces. | Use network policies to allow or deny traffic to pods. By default, all traffic is allowed between pods within a cluster. For improved security, define rules that limit pod communication. Network policy is a Kubernetes feature that lets you control the traffic flow between pods. You can choose to allow or deny traffic based on settings such as assigned labels, namespace, or traffic port. The use of network policies gives a cloud-native way to control the flow of traffic. |
| Network | Do you configure default network policies in each namespace (e.g. Deny all)? | Yes | N/A | Start by creating a deny all policy in each namespace and then add specific policies. |

### High Availability – Disaster Recovery

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Question** | **Answer** | **Comment** | **Recommendation** |
| HA-DR | Can you perform a whitespace deployment? | No | Mutua doesn't have plans for exercises like this.  ArgoCD is used but some requirements need to be automated (i.e. secret creation in an external repo). Recommendations for external secrets are provided in detail in the section [Whitespace Deployment – External Secrets](#_Whitespace_Deployment_–) | A whitespace (greenfield) deployment is the exercise to delete everything and to redeploy the whole platform in an automated way |
| HA-DR | Do you have defined Recovery Time Objective (RTO) per application on OCP Cluster? | No | It is not defined. | [RTO (Recovery Time Objective) Explained. It is recomended to have the RTO defined as part of the DR plan.](https://www.ibm.com/services/business-continuity/rto) |
| HA-DR | Do you have defined Recovery Point Objective (RPO) per application on OCP Cluster? | Yes | They don't have because all apps are mostly stateless. | Nonetheless, it is a good practice to define a RPO for Openshift, although it is not needed for the applications deployed in OCP. |
| HA-DR | How often do you perform a Disaster Recovery Exercise? | never | Mutua is in contact with Red Hat TAM to perform to implement this exercise. | Define a DR solution is highly recommended. |
| HA-DR | In case of a DR exercise do you achieve the defined RTO? | No | Not yet executed | [RTO (Recovery Time Objective) Explained](https://www.ibm.com/services/business-continuity/rto) |
| HA-DR | In case of a DR exercise do you achieve the defined RPO? | No | Not yet executed | [RPO (Recovery Point Objective) Explained](https://www.ibm.com/services/business-continuity/rpo) |
| HA-DR | Do you obtain backups for applications’ persistent volumes? If yes, how? | No | They are working in testing with Velero but it is not yet on production. | It is highly important to backup the PVs in case of a disaster. |
| HA-DR | How often do you test the recovery process of backups obtained? | never | Mutua have a daily etcd backup configured in Nagios. Mutua had to do a restore in February. | Restore of backups should be tested as frequently as possible |

## Red Hat Insights

### Workloads are still using the deprecated APIs which will be removed in the next release

OpenShift Container Platform 4.8.14 introduced a requirement that an administrator must provide a manual acknowledgment before the cluster can be upgraded from OpenShift Container Platform 4.8 to 4.9. In Mutua’s case only the Lab OCP cluster is in version 4.8, the production cluster is still in 4.6, so this warning only applies to the Lab cluster.  
  
Kubernetes 1.22 removed the following deprecated v1beta1 API which is used in Mutua and must be migrated to the appropriate new version before upgrading to OpenShift Container Platform 4.9

Resource: Ingress

API: extensions/v1beta1

* The extensions/v1beta1 and networking.k8s.io/v1beta1 API versions of Ingress is no longer served as of v1.22.
* Migrate manifests and API clients to use the networking.k8s.io/v1 API version, available since v1.19.
* All existing persisted objects are accessible via the new API

Notable changes:

* spec.backend is renamed to spec.defaultBackend
* The backend serviceName field is renamed to service.name
* Numeric backend servicePort fields are renamed to service.port.number
* String backend servicePort fields are renamed to service.port.name
* pathType is now required for each specified path. Options are Prefix, Exact, and ImplementationSpecific. To match the undefined v1beta1 behavior, use ImplementationSpecific.

## IBM Cloud Monitoring

During the 7 days period for monitoring the OCP cluster defined by IBM & Mutua, the following information was gathered. Unfortunately, the monitoring period had to be reduced to 2 days due to some issues with the Sysdig agent and OCP. But we managed to gather enough information to evaluate the performance and stability of the infrastructure. For our analysis, we emphasize on the information obtained in May 24th – May 25th.

### Cluster Capacity Planning

A screenshot of a computer

Description automatically generated

Available cluster resources can cover the load recorded during the period of assessment. The cluster has more capacity than what it is used, so additional workloads could be deployed in this cluster. Only 13% of the CPU and 30% of the memory is used. The applications request more CPU and memory than what they use.

### Limit overcommit

Graphical user interface, application

Description automatically generated

Taking under consideration the percentage of “Limit overcommit of CPU cores” and the actual number of overcommitted cores, it would be a good practice to re-evaluate if the limits defined can be reduced or verify for which applications the expected load may indeed reach the CPU limit set. On the other hand, the overcommit for memory is well handled.

A node is overcommitted when it has a pod scheduled that makes no request, or when the sum of limits across all pods on that node exceeds available machine capacity.

In an overcommitted environment, it is possible that the pods on the node will attempt to use more compute resource than is available at any given point in time. When this occurs, the node must give priority to one pod over another. The facility used to make this decision is referred to as a Quality of Service (QoS) Class.

For each compute resource, a container is divided into one of three QoS classes with decreasing order of priority:

|  |  |  |
| --- | --- | --- |
| **Priority** | **Class Name** | **Description** |
| 1 (highest) | **Guaranteed** | If limits and optionally requests are set (not equal to 0) for all resources and they are equal, then the container is classified as **Guaranteed**. |
| 2 | **Burstable** | If requests and optionally limits are set (not equal to 0) for all resources, and they are not equal, then the container is classified as **Burstable**. |
| 3 (lowest) | **BestEffort** | If requests and limits are not set for any of the resources, then the container is classified as **BestEffort**. |

Memory is an incompressible resource, so in low memory situations, containers that have the lowest priority are terminated first:

* **Guaranteed** containers are considered top priority, and are guaranteed to only be terminated if they exceed their limits, or if the system is under memory pressure and there are no lower priority containers that can be evicted.
* **Burstable** containers under system memory pressure are more likely to be terminated once they exceed their requests and no other **BestEffort** containers exist.
* **BestEffort** containers are treated with the lowest priority. Processes in these containers are first to be terminated if the system runs out of memory.

### Use of Allocated Resources

Calendar

Description automatically generated

The ratio of actual CPU and Memory utilization, versus the respective limits and requests, is significantly low. In general terms, the specific cluster can accommodate load much higher than the existing one.

### Unused Requested Resources (CPU / RAM)

Graphical user interface, application

Description automatically generatedThere is an issue with the requests defined for CPU and memory, there are 109 CPU cores and 80Gb memory reserved for applications that never use those resources, the issue comes from not well-adjusted requests in the applications deployed on Openshift. As mentioned before, the cluster has capability to handle bigger workloads, releasing unused CPU and memory would be good to accommodate new workloads.

### Container Golden Signals

A picture containing timeline

Description automatically generated

There is a high number of inbound HTTP Errors for production router, this is a situation that needs to be investigated further. Also, significant high network latency (~16 mins) is recorded with some pods at some points in time during the analysis, this is also a situation for further investigation.

### Node Resources Usage

Timeline

Description automatically generated

Graphical user interface, application

Description automatically generated

CPU load seems not be equally distributed among all worker nodes but instead, but significant load is requested in some development nodes, especially in node des8 the pods in that worker node are competing for CPU, reaching levels of 150% of CPU node resources. Besides the need to adjust requests and limits, some non-prod worker nodes may need resizing. Also, Mutua should reevaluate the use of pod affinity and anti-affinity rules as well as taints and tolerations. For further clarifications please refer Openshift Documentation available at <https://docs.openshift.com/container-platform/4.8/nodes/scheduling/nodes-scheduler-taints-tolerations.html> and <https://docs.openshift.com/container-platform/4.8/nodes/scheduling/nodes-scheduler-pod-affinity.html>.

### Network Latency per Pod

Text

Description automatically generated

Significant network latency is identified in several pods in production environment. This situation may indicate networking problems, applications performance issues, etc. It might be an isolated issue during the analysis, but it might be worthed to look into this situation.

### CPU & Memory used by applications vs request (Monitor applications)



A few of Mutua’s applications are using more CPU than what those application requested, in some extreme cases up 24000%, 9000%, etc. This data means incorrect requests & limits are defined for the applications, again a resource management issue appears. The recommendation is to adjust requests & limits.

# Key Recommendations

## Openshift Architecture

The following are key recommendations from an Openshift architecture point of view that IBM has assessed for Mutua Madrileña.

* At a minimum, Mutua should plan for two clusters - one for production, and one for non-production. Ideally, an Openshift cluster is recommended for each environment development, UAT and production. It is not a good practice to merge production and non-production environments in one cluster. The reason is for patching/updates/installation, this needs to be verified first on non-production before applying to production cluster. Maintenance is more difficult without non-production environments, which complicates rollout and rollback tasks.
* The % of used CPU versus the requests/limits is very low, reaching only 13% of the actual CPU capacity. Mutua’s Openshift cluster can accommodate more and bigger workloads. However, the values for limits and request need to be adjusted, because even using only 13% of CPU capacity, the cluster has 200% CPU over-commitment. This may lead to capacity issues for new deployments.
* The Openshift cluster subject to this analysis has 72 nodes, when the number of worker nodes increases to almost 100 worker nodes, it is important to study the impact in the master and infrastructure nodes. In case number of nodes is bigger than 100 then double the CPU and memory for the master and infrastructure nodes.
* As a rule of thumb, when the CPU capacity of a cluster is bigger than 48 CPUs, it is recommended to use worker nodes of 16 vCPUs instead of smaller worker nodes. In general, Mutua’s worker Nodes have 8 vCPUs, therefore, it is recommended to modify those worker nodes to a capacity of 16 vCPUs, this modification also has a positive impact in the resources needed for the master and infrastructure nodes, because Mutua will reduce the number of nodes from the actual 72.
* Mutua has an Openshift cluster with High Availability across two data centers, but there is no Disaster Recovery solution in place. It is highly recommended to study a Disaster Recovery solution. For example, a possible scenario for a DR solution could be making use of RH ACM to deploy OCP cluster and applications to a cloud provider or using Vmware technology to leverage a DR solution to a third data center or to a cloud provider, there several options that can be study. Along, with a possible DR solution, it is highly recommended to defined RPO and RTO for a possible disaster in the current Openshift cluster.
* Another important topic is that during the assessment some worker nodes dedicated for the development and UAT namespace environments nodes have been identified as undersized, also the requests/limits of deployments need to be adjusted. Mutua could run into deployment issues for new applications in both environments.

## Openshift Security

From the security point of view related to Openshift, some good practices have been identified that Mutua is already applying. Nonetheless, some improvements areas have been identified.

**Good security practices identified in Mutua**

* Network Policis are been used to isolate namespaces.
* Mutua is using Imperva as a WAF to securely expose applications
* Mutua is working in deploying PrismaCloud to provide security analysis of images and a security framework for containers.
* The access to the Openshift APIs and console is only possible from the bastion server.
* Internet access to containers is blocked by default, any access to internet needs from the containers needs to be approved an opened under request. This avoids any security issue to exposing data or access to malware.

**Possible security improvements related to Openshift**

* Applications use the default Security Context Constrains (SCC) defined by Openshift. Mutua can defined their own customs SCCs provides only the minimum set of permissions to execute the applications.
* Non-prod and Prod environments in the same Openshift cluster is a potential security issue. Although Mutua is using networking and network policies to isolate the namespaces defined for the environments, non-prod and prod environments are part of the same Openshift cluster, master and infrastructure nodes have access to all namespaces, therefore, there is a possible way to exploit this situation if a future security breach is discovered.
* Periodically assess Openshift Role-based access control (RBAC) assigned to the Openshift users to ensure a correct security definition.
* Recommended adoption of Security Software Development Lifecycle (sSDL) adding tools for static and dynamic code analysis, and code quality tools, like sonarQube.
* Use RH ACM to deploy policies to enforce security in the Openshift clusters. In the following link there are ready-to-deploy policies with RH ACM <https://github.com/stolostron/policy-collection/tree/main/stable>

## Resiliency Tests

It is indispensable to ensure that a system/service built is able to withstand chaotic conditions as failures are inevitable. Chaos engineering helps in boosting confidence in a system's resilience by “breaking things on purpose.” While it may seem counterintuitive, it is crucial to deliberately inject failures into a complex system like OpenShift/Kubernetes and check whether the system recovers gracefully without any downtime and doesn’t suffer in terms of performance and scalability. Chaos engineering is a discipline to identify potential problems and enhance the system’s resilience.

**Kraken to the Rescue**

Red Hat developed a chaos tool named Kraken with the aim of “breaking things on purpose” and identifying future issues. Kraken enables the user to effortlessly inject chaos in a Kubernetes/OpenShift cluster. The user can continuously cause chaos and watch how the cluster responds to various failure injections over a long run. Additionally, one can validate if the cluster completely recovers from chaos and returns to its normal healthy state after a single set of failure injections.

**Kubernetes/OpenShift chaos scenarios supported**

* Pod Scenarios
* Container Scenarios
* Node Scenarios
* Time Scenarios
* Litmus Scenarios
* Cluster Shut Down Scenarios
* Namespace Scenarios
* Zone Outage Scenarios
* Application outages
* PVC scenario
* Network Chaos

For further information, please visit <https://github.com/cloud-bulldozer/kraken>

Diagram

Description automatically generated

## Whitespace Deployment – External Secrets

External Secrets Operator is a Kubernetes operator that integrates external secret management systems like IBM Secret Manager, HashiCorp Vault, Gitlab and many more. The operator reads information from external APIs and automatically injects the values into a Kubernetes Secret.

The goal of External Secrets Operator is to synchronize secrets from external APIs into Kubernetes. ESO is a collection of custom API resources - ExternalSecret, SecretStore and ClusterSecretStore that provide a user-friendly abstraction for the external API that stores and manages the lifecycle of the secrets for you.

Diagram

Description automatically generated

For further information, please visit

* [https://external-secrets.io](https://external-secrets.io/)
* <https://github.com/external-secrets/kubernetes-external-secrets>

# IBM Next Step Proposals

## IBM Workshop and Education enablement sessions

IBM CSM team delivers education and enablement sessions and workshops for IBM clients. These sessions are free of charge for our clients. For Mutua we propose to deliver a 3 days **Openshift** Workshop, with a special focus on applications integration with Tekton Pipelines vs Jenkins vs S2I [https://cloudnativetoolkit.dev/](https://cloudnativetoolkit.dev/ )

This workshop is a hands-on workshop where different technologies related to applications modernization are explained and the attendees can do labs to test and gain a better understanding of the technology. The workshop covers technologies like Docker, Kubernetes, Openshift and DevOps, it is also part of the workshop the topic of application modernization, for that purpose IBM Transformation Advisor is also explained, this tool will help IBM clients in the transformation of current applications to a containerized version.



**Introductory session to GitOps**

One of the current trends in IT management infrastructure is GitOps, which can be considered an evolution in Infrastructure as Code. GitOps is an operational framework that takes DevOps best practices used for application development such as version control, collaboration, compliance, and CI/CD, and applies them to infrastructure automation.

GitOps delivers:

* A standard workflow for application development
* Increased security for setting application requirements upfront
* Improved reliability with visibility and version control through Git
* Consistency across any cluster, any cloud, and any on-premise environment

GitOps can be used to build development pipelines, code applications, manage configurations, provision Kubernetes clusters, and deploy on Kubernetes or container registries.

Diagram

Description automatically generated

<https://www.redhat.com/en/topics/devops/what-is-gitops>

<https://production-gitops.dev/>

Moreover, Red Hat Advanced Cluster Management (RH ACM), which Mutua has already deployed, complements perfectly the GitOps approach, to include the management of applications deployments across a multi cluster environment.

## Application Resource Management - Turbonomics

Probably the most important finding in the Openshift Assessment in Mutua is the resource management situation, as explained in previous sections, Mutua is using only 13% of the available CPU, but the limits of the applications have an unnecessary high value, same situation happens for the request of some applications while other applications don’t have the necessary minimum value for the request. Moreover, there are nodes overloaded while there is available capacity.

Application resource management is not a simple and easy task, IT administrators don’t have the necessary visibility and time to do it and development team doesn’t have a clear picture of the expected workload, so the tendency is to dedicate more resources than what is actually needed. The main issue is that resource management is a complex problem to be solved by humans in an effective way.

To resolve the resource management problem, we need an application resource management (ARM) system. The objective of an ARM is to move from a static allocation model to a dynamic consumption-based model. In this model, discovery of a client’s full application stack is automatic and continuous, with real time performance analysis. Turnomics integrates with many APM systems, like Dynatrace and many others .Allocation of resources is automatic based on the performance analysis and ai-driven recommendations.

Turbonomic begins by ingesting all observable metrics across the entire landscape and up and down the full stack including storage, physical and virtual servers, containers and pods, memory, CPU and network resources. Before allocating resources, policies are checked to ensure Turbonomic adheres to them.

Graphical user interface

Description automatically generated

## Modernization in Openshift of Legacy Applications in WebSphere

Mutua's applications deployed in WebSphere can be modernized to containerized applications thanks to the Cloud Pak for Applications entitlements that provides unrestricted Openshift entitlements and access to Red Hat and IBM runtimes. Mutua Madrileña currently has 152 VPCs of Cloud Paks for Applications, which translate into 1:1 core entitlement of Openshift; this means that Mutua is capable of transforming current deployments of WebSphere into Openshift, without additional cost.

Included in Cloud Paks for Applications, IBM provides Transformation Advisor a tool that can be used to analyse legacy applications deployed in WebSphere and understand the estimated cost to modernize. The tool provides a detailed analysis of the actions needed to transform legacy applications into containers.



Graphical user interface, text

Description automatically generated

## Cloud Pak for Multicloud Management – Red Hat Advanced Cluster Management for Kubernetes.

Organizations with multiple Openshift Container Platform Clusters deployed, come to the need of centrally managing them. Taking under consideration that Mutua already has deployed **Cloud Pak for Multicloud Management**, **Red Hat Advanced Cluster Management for Kubernetes** can be of use.

The benefits of RHACM can be described as following:

**Unified multicluster management**

* Centrally create, update, and delete Kubernetes clusters across multiple private and public clouds
* Search, find, and modify any Kubernetes resource across the entire domain
* Quickly troubleshoot and resolve issues across your federated domain
* When creating or updating clusters, automate tasks such as configuring cloud-defined storage, static IP addresses, updating network components (like firewalls or load balancers), and more with the integration of Red Hat Ansible Automation Platform.

**Policy based governance, risk and compliance**

* Centrally set and enforce policies for security, applications, and infrastructure. <https://github.com/stolostron/policy-collection/tree/main/stable>
* Quickly visualize detailed auditing on configuration of apps and clusters
* Immediate visibility into your compliance posture based on your defined standards
* Automate remediation of policy violations and gather audit information about the clusters for analysis with the integration of Red Hat Ansible Automation Platform

**Advanced application life-cycle management**

* Define and deploy applications across clusters based on policy
* Quickly view service endpoints and pods associated with your application topology—with all the dependencies
* Automatically deploy applications to specific clusters based on channel and subscription definitions
* When deploying or updating applications, automate configurations like networking, databases, and more with the integration of Red Hat Ansible Automation Platform

**Multicluster observability for health and optimization**

* Get an overview of multicluster health and optimization using out-of-the-box multicluster dashboards with the ability to store long-term data
* Easily sort, filter, and do a deep scan of individual clusters or, at the aggregated multicluster level
* Get an aggregated view of cluster metrics
* Troubleshoot faster using the Dynamic Search and Visual Web Terminal capabilities

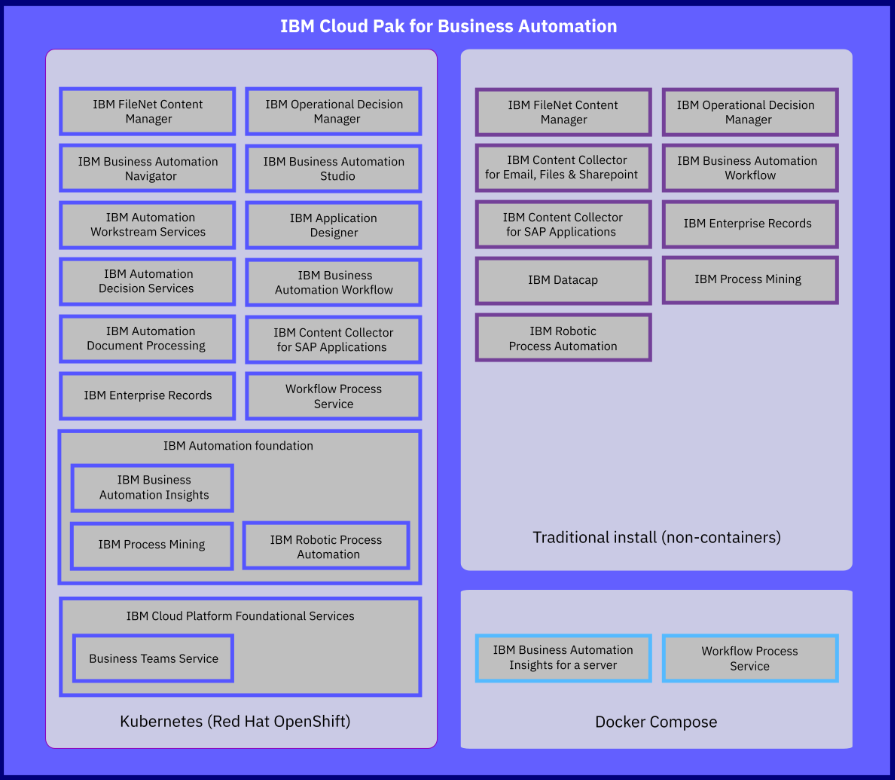
## Modernization of IBM products into Openshift

Nowadays, many organizations are involved in a transformation and modernization due to the new computing paradigms and the inclusion of Cloud providers as part of the organization’s infrastructure. As part of this transformation, many products are been migrated to a containerized environment, which is ideal for the HA/DR scenarios when incluing Cloud providers in the organization’s infrastructure. Deploying products in a containerized environment allows an easy migration among cloud providers and more flexibility.

IBM has a containerized version of many of their products, specially those included under the Cloud Paks umbrella. If Mutua’s technology strategy includes the contanerization of products, it is important to highlight that IBM products that Mutua is already using like FileNet, MDM or API Connect, have a containerized version capable to be deployed into Kubernetes environments.

Among those Kubernetes environments Openshift takes a important role, those products included in the Cloud Paks have some advantages when deployed in Openshift; as part of the license free of charge amount of Openshift entitlements are provided to deploy the products, moreover, some storage solutions licenses are also provideded, solutions like Openshift Data Foundation, IBM Spectrum Scale, etc. This free entitlements are restricted for the use and deployment of the IBM product.

As an example, in the image below, it is showed the products included in the Cloud Pak for Business Automation, which of them have a containers version in the left side and which of them have a traditional deployment version. For instance, FileNet has both versions available to be used by IBM clients.



1. For further information please visit <https://docs.openshift.com/container-platform> [↑](#footnote-ref-1)