Chapter 1.  Introduction to Kubernetes and OpenShift

Containers and Kubernetes

Objectives

* Describe the main characteristics of containers and Kubernetes.

Introduction to Containers and Kubernetes

Red Hat OpenShift Container Platform (RHOCP) is a complete platform to run applications in clusters of servers. OpenShift is based on existing technologies such as *containers* and Kubernetes. Containers provide a way to package applications and their dependencies that can be executed on any system with a container runtime.

Kubernetes provides many features to build clusters of servers that run containerized applications. However, Kubernetes does not intend to provide a complete solution, but rather provides extension points so system administrators can complete Kubernetes. OpenShift builds on the extension points of Kubernetes to provide a complete platform.

Containers Overview

A *container* is a process that runs an application independently from other containers on the host. Containers are created from a *container image*, which includes all the runtime dependencies of the application. Containers can then be executed on any host, without requiring the installation of any dependency on the host, and without conflicts between the dependencies of different containers.

A screenshot of a computer

Description automatically generated

Figure 1.1: Applications in containers versus on the host operating system

Containers use Linux kernel features, such as namespaces and control groups (cgroups). For example, containers use cgroups for resource management, such as CPU time allocation and system memory. Namespaces isolate a container's processes and resources from other containers and the host system.

The Open Container Initiative (OCI) maintains standards about containers, container images, and container runtimes. Because most container engine implementations are designed to conform to the OCI specifications, applications that are packaged according to the specification can run on any conforming platform.

Kubernetes Overview

Kubernetes is a platform that simplifies the deployment, management, and scaling of containerized applications.

You can run containers on any Linux system by using tools such as Podman. Further work is required to create containerized services that, for example, can run across a cluster for redundancy.

Kubernetes creates a cluster that runs applications across multiple nodes. If a node fails, then Kubernetes can restart an application in a different node.

Kubernetes uses a declarative configuration model. Kubernetes administrators write a definition of the workloads to execute in the cluster, and Kubernetes ensures that the running workloads match the definition. For example, an administrator defines several workloads. Each workload defines the amount of required memory. Kubernetes then creates the necessary containers in different nodes, to meet the memory requirements.

Kubernetes defines workloads in terms of *pods*. A pod is one or multiple containers that run in the same cluster node. Pods with multiple containers are useful in certain situations, when two containers must run in the same cluster node to share some resource. For example, a job is a workload that runs a task in a pod until completion.

Kubernetes Features

Kubernetes offers the following features on top of a container infrastructure:

**Service discovery and load balancing**

Distributing applications over multiple nodes can complicate communication between applications.

Kubernetes automatically configures networking and provides a DNS service for pods. With these features, pods can communicate with services from other pods transparently across nodes by using only hostnames instead of IP addresses. Multiple pods can back a service for performance and reliability. For example, Kubernetes can evenly split incoming requests to an NGINX web server by considering the availability of the NGINX pods.

**Horizontal scaling**

Kubernetes can monitor the load on a service, and create or delete pods to adapt to the load. With some configurations, Kubernetes can also provision nodes dynamically to accommodate cluster load.

**Self-healing**

If applications declare health check procedures, then Kubernetes can monitor, restart, and reschedule failing or unavailable applications. Self-healing protects applications both from internal failure (the application stops unexpectedly) or external failure (the node that runs the application becomes unavailable).

**Automated rollout**

Kubernetes can gradually roll out updates to your application's containers. If something goes wrong during the rollout, then Kubernetes can roll back to the previous version of the deployment. Kubernetes routes requests to the rolled out version of the application, and deletes pods from the previous version when the rollout completes.

**Secrets and configuration management**

You can manage the configuration settings and secrets of your applications without requiring changes to containers. Application secrets can be usernames, passwords, and service endpoints, or any configuration setting that must be kept private.

**Note**

Kubernetes does not encrypt secrets.

**Operators**

Operators are packaged Kubernetes applications that can manage Kubernetes workloads in a declarative manner. For example, an operator can define a database server resource. If the user creates a database resource, then the operator creates the necessary workloads to deploy the database, configure replication, and back up automatically.

Kubernetes Architectural Concepts

Kubernetes uses several servers (also called *nodes*) to ensure the resilience and scalability of the applications that it manages. These nodes can be physical or virtual machines. The nodes come in two types, each of which provides a different aspect of the cluster operations.

*Control plane* nodes provide global cluster coordination for scheduling the deployed workloads. These nodes also manage the state of the cluster configuration in response to cluster events and requests.

*Compute plane* nodes run the user workloads.

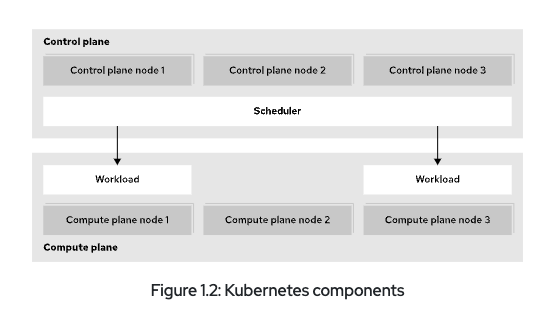


Figure 1.2: Kubernetes components

Although a server can act as both a control plane node and a compute node, the two roles are usually separated for increased stability, security, and manageability. Kubernetes clusters can range from small single-node clusters, to large clusters with up to 5,000 nodes.

The Kubernetes API and Configuration Model

Kubernetes provides a model to define the workloads to run on a cluster.

Administrators define workloads in terms of resources.

All resource types work in the same way, with a uniform API. Tools such as the kubectl command can manage resources of all types, even custom resource types.

Kubernetes can import and export resources as text files. By working with resource definitions in text formats, administrators can describe their workloads instead of performing the right sequence of operations to create them. This approach is called *declarative resource management*.

A screenshot of a computer program

Description automatically generated

Figure 1.3: Declarative resource management

Declarative resource management can reduce the work to create workloads and improve maintainability. Additionally, when using text files to describe resources, administrators can use generic tools such as version control systems to gain further benefits such as change tracking.

To support declarative resource management, Kubernetes uses *controllers* that continuously track the state of the cluster and perform the necessary steps to keep the cluster in the intended state. This process means that changes to resources often require some time to be effective. However, Kubernetes can automatically apply complex changes. For example, if you increase the RAM requirements of an application that cannot be satisfied on the current node, then Kubernetes can move the application to a node with sufficient RAM. Kubernetes redirects traffic to the new instance only when the movement is complete.

Kubernetes also supports *namespaces*. Administrators can create namespaces, and most resources must be created inside a namespace. Besides helping organize large amounts of resources, namespaces provide the foundations for features such as resource access. Administrators can define permissions for namespaces, to allow specific users to view or modify resources.

Red Hat OpenShift Components and Editions

Objectives

* Describe the relationship between OpenShift, Kubernetes, and other Open Source projects, and list key features of Red Hat OpenShift products and editions.

Introduction to Red Hat OpenShift

Kubernetes provides many features to run container workloads on clusters. However, for some features, Kubernetes provides only the building blocks to implement them, because different environments might need different solutions. Kubernetes administrators can select an existing solution or implement their own solution to fit their specific requirements.

OpenShift uses the extensibility of Kubernetes to build a complete solution by adding the following features to a Kubernetes cluster:

**Integrated developer workflow**

When running applications on Kubernetes, you need to build and store container images for the applications. OpenShift integrates a built-in container registry, CI/CD pipelines, and S2I, a tool to build artifacts from source repositories to container images.

**Observability**

To achieve the intended reliability, performance, and availability of applications, cluster administrators might need additional tools to prevent and solve issues. OpenShift includes monitoring and logging services for both your applications and the cluster.

**Server management**

Kubernetes requires an operating system to run on that must be installed, configured, and maintained.

OpenShift provides installation and update procedures for many scenarios.

Additionally, hosts in a cluster use Red Hat Enterprise Linux CoreOS (RHEL CoreOS) as the underlying operating system. RHEL CoreOS is an immutable operating system that is optimized for running containerized applications. RHEL CoreOS uses the same kernel and packages as Red Hat Enterprise Linux. OpenShift also provides features to manage RHEL CoreOS following the Kubernetes configuration model.

OpenShift also brings unified tools and a graphical web console to manage all the different capabilities, and additional enhancements such as improved security measures.

This diagram shows many of the projects that provide the various functions within each OpenShift cluster:

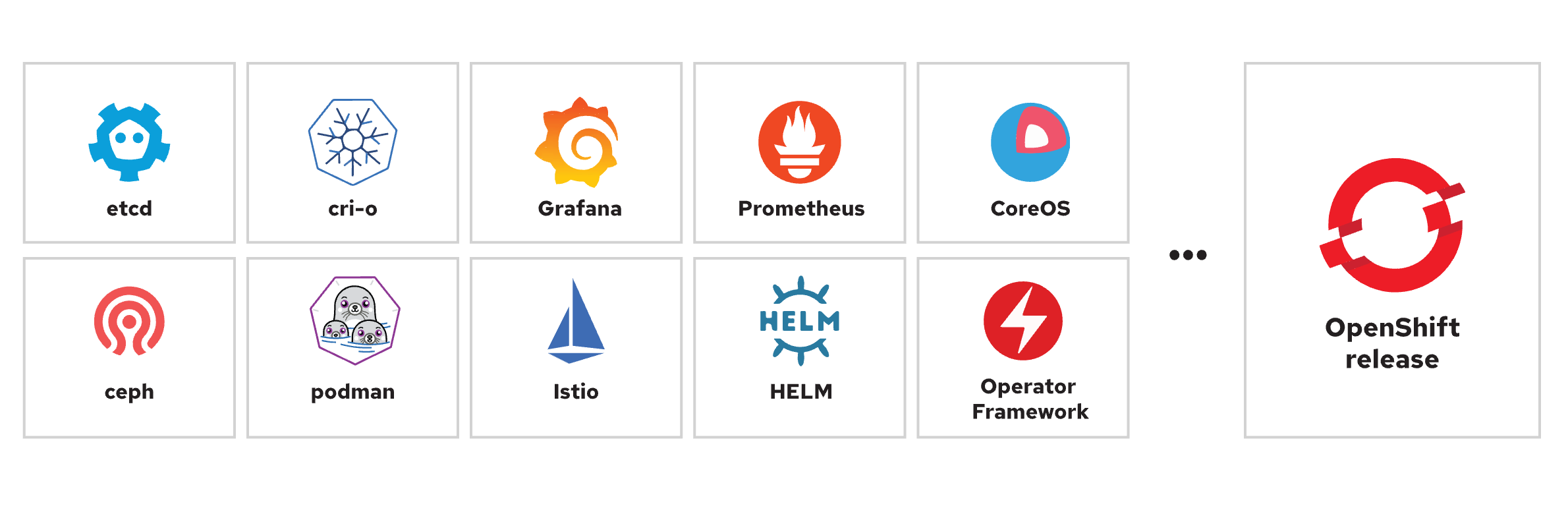


Figure 1.4: Open source projects in an OpenShift release

Because OpenShift features build on Kubernetes extensibility, administrators can often operate these features by using their existing Kubernetes knowledge. Besides Kubernetes knowledge, OpenShift administrators can use most existing products that are designed for Kubernetes.

Red Hat OpenShift Editions

When you initially explore OpenShift, using Red Hat OpenShift Local is a viable approach that deploys a cluster on a local computer for testing and exploration. Red Hat also provides a Developer Sandbox where you get 30 days of free access to a shared OpenShift cluster. These options provide access to a cluster and support testing and exploration as you consider adopting OpenShift, but are not suitable environments for production deployments.

When you are ready to adopt Red Hat OpenShift for production workloads, various editions are available to suit any business requirement for the cluster deployment.

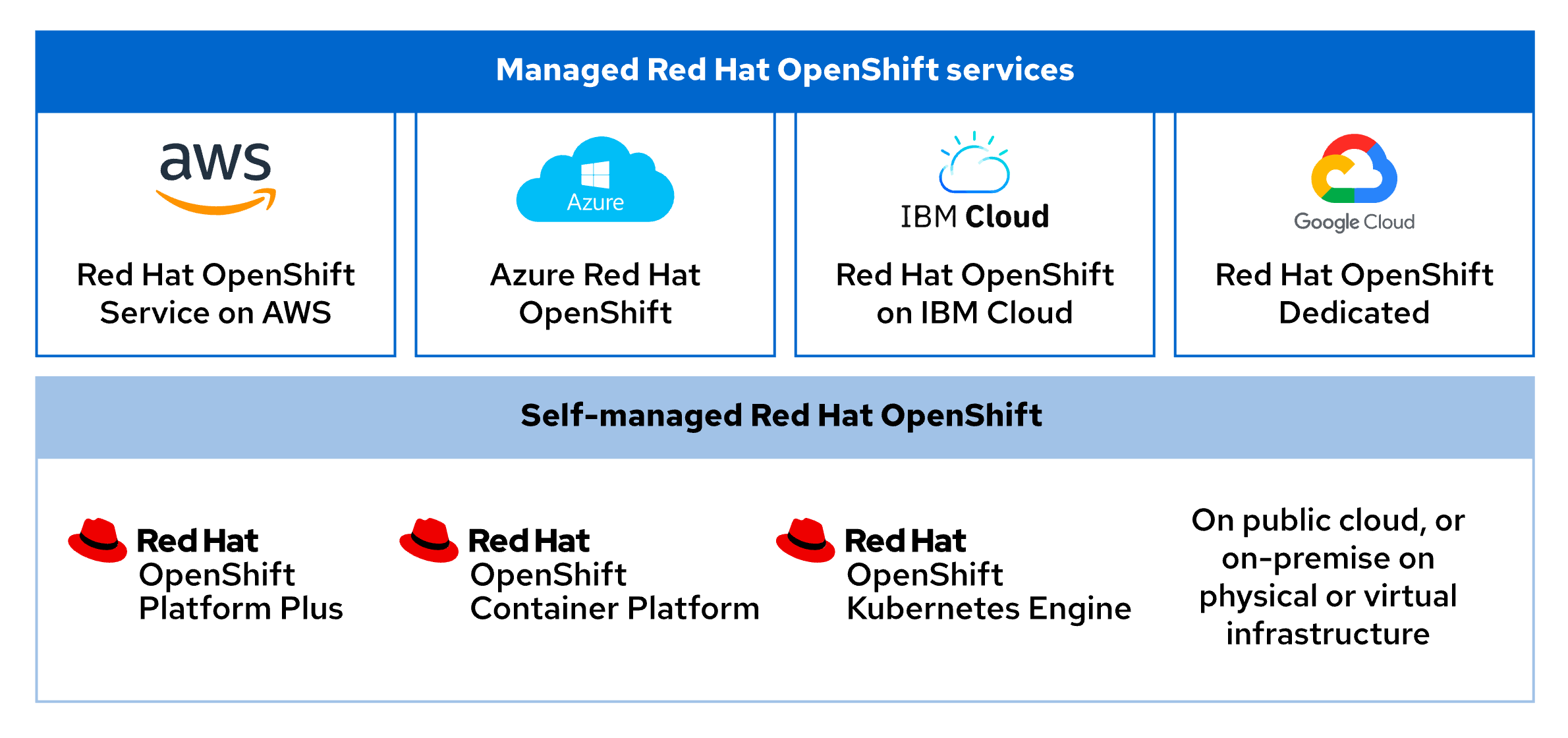


Figure 1.5: Product editions

Public cloud partners, such as Amazon Web Services, Microsoft Azure, IBM Cloud, and Google Cloud each provide quick access to an on-demand Red Hat OpenShift deployment. These managed deployments offer quick access to a cluster on infrastructure that you can rely on from a Red Hat trusted cloud provider.

You can also deploy a Red Hat OpenShift cluster by using the available installers on physical or virtual infrastructure, either on-premise or in a public cloud. These self-managed offerings are available in several forms.

The choice of deployment methods depends on many factors. When using the managed services, more responsibilities are delegated to Red Hat and the cloud provider. Because the installation process integrates with the cloud provider, the managed service creates and manages all necessary cloud resources.

When using the installers, you can still delegate hardware management to a cloud provider, or use your own. However, you manage the rest of the solution. With self-managed editions, you have greater control and flexibility, but you also take on greater responsibilities over the service.

For example, with managed services, the Red Hat Site Reliability Engineering teams update the clusters and remedy update issues (although you still participate in scheduling the updates). On self-managed editions, you update the clusters and remedy update issues. On the other hand, on self-managed editions you have complete control over aspects such as authentication, when managed editions might restrict some options.

Each managed edition documents the responsibilities for customers, Red Hat, and the cloud provider.

Additionally, to assist in managing clusters, the Red Hat Insights Advisor is available from the Red Hat Hybrid Cloud Console. The Insights Advisor helps administrators to identify and remediate cluster issues by analyzing data that the Insights Operator provides. The data from the operator is uploaded to the Red Hat Hybrid Cloud Console, where you further inspect the recommendations and their impact on the cluster.

The contents of this course apply to both managed services and self-managed editions.

Red Hat OpenShift Kubernetes Engine includes the latest version of the Kubernetes platform with the additional security hardening and enterprise stability that Red Hat is famous for delivering. This deployment runs on the Red Hat Enterprise Linux CoreOS immutable container operating system, by using Red Hat OpenShift Virtualization for virtual machine management, and provides an administrator console to aid in operational support.

Red Hat OpenShift Container Platform builds on the features of the OpenShift Kubernetes Engine to include additional cluster manageability, security, stability, and ease of application development for businesses. Additional features of this tier include a developer console, as well as log management, cost management, and metering information. This offering adds Red Hat OpenShift Serverless (Knative), Red Hat OpenShift Service Mesh (Istio), Red Hat OpenShift Pipelines (Tekton), and Red Hat OpenShift GitOps (Argo CD) to the deployment.

Red Hat OpenShift Platform Plus expands further on the offering to deliver the most valuable and robust available features. This offering includes Red Hat Advanced Cluster Management for Kubernetes, Red Hat Advanced Cluster Security for Kubernetes, and the Red Hat Quay private registry platform. For the most complete and full-featured container experience, Red Hat OpenShift Platform Plus bundles all the necessary tools for a complete development and administrative approach to containerized application platform management.

All OpenShift editions use the same code. Most content in this course applies to all OpenShift editions.

Overview of the Red Hat OpenShift Web Console

The Red Hat OpenShift Web Console provides a graphical user interface to perform many administrative tasks for managing a cluster. The web console uses the Kubernetes APIs and OpenShift extension APIs to deliver a robust graphical experience. The menus, tasks, and features within the web console are always available by using the CLI. The web console provides ease of access and management for the complex tasks that cluster administration requires.

Kubernetes provides a web-based dashboard, which is not deployed by default within a cluster. The Kubernetes dashboard provides minimal security permissions, and accepts only token-based authentication. This dashboard also requires a proxy setup that limits access to the web console from only the system terminal that creates the proxy. By contrast with the stated limitations of the Kubernetes web console, OpenShift includes a fuller-featured web console.

The OpenShift web console is not related to the Kubernetes dashboard, but is a separate tool for managing OpenShift clusters. Additionally, operators can extend the web console features and functions to include more menus, views, and forms to aid in cluster administration.

Accessing the OpenShift Web Console

You access the web console by any modern web browser. The web console URL is generally configurable, and you can discover the address for your cluster web console by using the oc command-line interface (CLI). From a terminal, you must first authenticate to the cluster via the CLI by using the oc login -u <USERNAME> -p <PASSWORD> <API\_ENDPOINT>:<PORT> command:

[user@host ~]$ **oc login -u developer -p developer https://api.ocp4.example.com:6443**

Login successful.

*...output omitted...*

Then, you execute the oc whoami --show-console command to retrieve the web console URL:

[user@host ~]$ **oc whoami --show-console**

https://console-openshift-console.apps.ocp4.example.com

Lastly, use a web browser to navigate to the URL, which displays the authentication page:

|  |
| --- |
| A screenshot of a login box  Description automatically generated |

Figure 1.6: The OpenShift authentication page

Using the credentials for your cluster access brings you to the home page for the web console.

A screenshot of a computer

Description automatically generated

Figure 1.7: The OpenShift home page

Web Console Perspectives

The OpenShift web console provides the Administrator and Developer console perspectives. The sidebar menu layout and the features that it displays differ between using the two console perspectives. The first item on the console sidebar menu is the perspective switcher, to switch between the Administrator and the Developer perspectives.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

Figure 1.8: The OpenShift web console perspective switcher

Each perspective presents the user with different menu categories and pages that cater to the needs of the two separate personas. The Administrator perspective focuses on cluster configuration, deployments, and operations of the cluster and running workloads. The Developer perspective pages focus on creating and running applications.

OpenShift Web Console Layout

From the web console home page, the primary navigation method is through the sidebar. The sidebar organizes cluster functions and administration into several major categories. By selecting a category from the sidebar, the category expands to reveal the various areas that each provide specific cluster information, configuration, or functionality.

**Note**

An initial login to the web console presents the option for a short informational tour. Click **Skip Tour** if you prefer to dismiss the tour option at this time.

A screenshot of a computer

Description automatically generated

By default, the console displays the **Home** → **Overview** page, which provides a quick glimpse of initial cluster configurations, documentation, and general cluster status. Navigate to **Home** → **Projects** to list all projects in the cluster that are available to the credentials in use.

You might initially peruse the **Operators** → **OperatorHub** page, which provides access to the collection of operators that are available for your cluster.

A screenshot of a computer

Description automatically generated

Figure 1.10: The OpenShift OperatorHub

By adding operators to the cluster, you can extend the features and functions that your OpenShift cluster provides. Use the search filter to find the available operators to enhance the cluster and to supply the OpenShift aspects that you require.

By clicking the link on the Operator Hub page, you can peruse the Developer Catalog.

A screenshot of a computer

Description automatically generated

Select any project, or use the search filter to find a specific project, to visit the Developer Catalog for that project, where shared applications, services, event sources, or source-to-image builders are available.

A screenshot of a web page

Description automatically generated

Figure 1.12: The Developer Catalog

After finding the preferred additions for a project, a cluster administrator can further customize the content that the catalog provides. By adding the necessary features to a project from this approach, developers can customize features to provide an ideal application deployment.

Red Hat OpenShift Key Concepts

When you navigate the OpenShift web console, it is useful to know some introductory OpenShift, Kubernetes, and container terminology. The following list includes some basic concepts that can help you to navigate the OpenShift web console.

* Pods: The smallest unit of a Kubernetes-managed containerized application. A pod consists of one or more containers.
* Deployments: The operational unit that provides granular management of a running application.
* Projects: A Kubernetes namespace with additional annotations that provide multitenancy scoping for applications.
* Routes: Networking configuration to expose your applications and services to resources outside the cluster.
* Operators: Packaged Kubernetes applications that extend cluster functions.

These concepts are covered in more detail throughout the course. You can find these concepts throughout the web console as you explore the features of an OpenShift cluster from the graphical environment.

Monitor an OpenShift Cluster

Objectives

* Navigate the Events, Compute, and Observe panels of the OpenShift web console to assess the overall state of a cluster.

Overview of Nodes, Machines, and Machine Configurations

In Kubernetes, a *node* is any single system in the cluster where pods can run. These systems are any of the bare metal, virtual, or cloud computers that are members of the cluster. Nodes run the necessary services to communicate within the cluster, and receive control plane operational requests. When you deploy a pod, an available node is tasked with satisfying the request.

Whereas the *node* and *machine* terms are often interchangeable, Red Hat OpenShift Container Platform (RHOCP) uses the *machine* term more specifically. In OpenShift, a *machine* is the resource that describes a cluster node. Using a machine resource is particularly valuable when using public cloud providers to provision infrastructure.

A MachineConfig resource defines the initial state and any changes to files, services, operating system updates, and critical OpenShift service versions for the kubelet and cri-o services. OpenShift relies on the Machine Config Operator (MCO) to maintain the operating systems and configuration of the cluster machines. The MCO is a cluster-level operator that ensures the correct configuration of each machine. This operator also performs routine administrative tasks, such as system updates. This operator uses the machine definitions in a MachineConfig resource to continually validate and remediate the state of cluster machines to the intended state. After a MachineConfig change, the MCO orchestrates the execution of the changes for all affected nodes.

**Note**

The orchestration of MachineConfig changes through the MCO is prioritized alphabetically by zone, by using the topology.kubernetes.io/zone node label.

Identifying Errors from Nodes

Administrators routinely view the logs and connect to the nodes in the cluster by using a terminal. This technique is necessary to manage a cluster and to remediate issues that arise. From the web console, navigate to **Compute** → **Nodes** to view the list of all nodes in the cluster.

A screenshot of a computer

Description automatically generated

Figure 1.41: Node list in the web console

Click a node's name to navigate to the overview page for the node. On the node overview page, you can view the node logs or connect to the node by using the terminal.

A screenshot of a computer

Description automatically generated

Figure 1.42: Logs page in the web console

From the previous page in the web console, view the node logs and investigate the system information to aid troubleshooting and remediation for node issues.

A screen shot of a computer

Description automatically generated

Figure 1.43: Terminal shell in the web console

The preceding page shows the web console terminal that is connected to the cluster node. From this tab, you can access the debug pod and use the commands from the host binaries to view the status of the node's services. An OpenShift node debug pod is an interface to a container that runs on the node.

Although making changes directly on the cluster node from the terminal is not recommended, it is common practice to connect to the cluster node for diagnostic investigation and remediation. From this terminal, you can use the same binaries that are available within the cluster node itself.

Additionally, the tabs on the node overview page show metrics, events, and the node's YAML definition file.

Accessing Pod Logs

Administrators often peruse pod logs to assess the health of a deployed pod or to troubleshoot pod deployment issues. Navigate to the **Workloads** → **Pods** page to view the list of all pods in the cluster.

A screenshot of a computer

Description automatically generated

Figure 1.44: Pods page in the web console

You can filter and order pods by project and by other fields. To view the pod details page, click a pod name in the list.

A screenshot of a computer

Description automatically generated

Figure 1.45: Pod details page

The pod details page contains links to pod metrics, environment variables, logs, events, a terminal, and the pod's YAML definition. The pod logs are available on the **Pods** → **Logs** page and provide information about the pod status. The **Pods** → **Terminal** page opens a shell connection to the pod for inspection and issue remediation. Although it is not recommended to alter a running pod, the terminal is useful for diagnosing and remediating pod issues. To fix a pod, update the pod configuration to reflect the necessary changes, and redeploy the pod.

Red Hat OpenShift Container Platform Metrics and Alerts

In an RHOCP cluster, HTTP service endpoints provide data metrics that are collected to provide information for monitoring cluster and application performance. These metrics are authored at the application level for each service by using the client libraries that are provided by Prometheus, an open source monitoring and alerting toolkit. Metrics data is available from the service /metrics endpoint. You can use the data for creating monitors to alert based on degradation of the service. Monitors are processes that continuously assess the value for a specific metric and provide alerts that are based on a predefined condition, to signal a degradation in the service or a performance issue. Authoring a ServiceMonitor resource defines how a specific service uses the metrics to define a monitor and the alerting values. The same approach is available for monitoring pods by defining a PodMonitor resource that uses the metrics that are gathered from the pod.

Depending on the monitor definitions, alerting is then available based on the metric that is polled and the defined success criteria. The monitor continuously compares the gathered metric, and creates an alert when the success criteria are no longer met. As an example, a web service monitor polls on the listening port, port 80, and alerts only if the response from that port becomes invalid.

From the web console, navigate to **Observe** → **Metrics** to visualize gathered metrics by using a Grafana-based data query utility. On this page, users can submit queries to build data graphs and dashboards, which administrators can view to gather valuable statistics for the cluster and applications.

For configured monitors, visit **Observe** → **Alerting** to view firing alerts, and filter on the alert severity to view those alerts that need remediation. Alerting data is a key component to help administrators to deliver cluster and application accessibility and functions.

Kubernetes Events

Administrators are typically familiar with the contents of log files for services, whereas logs tend to be highly detailed and granular. *Events* provide a high-level abstraction to log files and to provide information about more significant changes. Events are useful in understanding the performance and behavior of the cluster, nodes, projects, or pods, at a glance. Events provide details to understand general performance and to highlight meaningful issues. Logs provide a deeper level of detail for remediating specific issues.

The **Home** → **Events** page shows the events for all projects or for a specific project. You can further filter and search events.

A screenshot of a computer

Description automatically generated

Figure 1.46: RHOCP Events console

Red Hat OpenShift Container Platform API Explorer

Starting from version 4, RHOCP includes the API Explorer feature, for users to view the catalog of Kubernetes resource types that are available within the cluster. By navigating to **Home** → **API Explorer**, you can view and explore the details for resources. Such details include the description, schema, and other metadata for the resource. This feature is helpful for all users, and especially for new administrators.

A screenshot of a computer

Description automatically generated

Figure 1.47: The RHOCP API Explorer

Guided Exercise: Monitor an OpenShift Cluster

Assess the overall status of an OpenShift cluster by using the web console, and identify projects and pods of core architectural components of Kubernetes and OpenShift.

**Outcomes**

* Explore and show the monitoring features and components.
* Explore the Overview page to inspect the cluster status.
* Use a terminal connection to the master01 node to view the crio and kubelet services.
* Explore the Monitoring page, alert rule configurations, and the etcd service dashboard.
* Explore the events page, and filter events by resource name, type, and message.

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command ensures that the cluster is prepared for the exercise.

[student@workstation ~]$ **lab start intro-monitor**

**Instructions**

1. As the developer user, locate and then navigate to the Red Hat OpenShift web console.
   1. Use the terminal to log in to the OpenShift cluster as the developer user with the developer password.
   2. [student@workstation ~]$ **oc login -u developer -p developer \**
   3. **https://api.ocp4.example.com:6443**

*...output omitted...*

* 1. Identify the URL for the OpenShift web console.api.ocp4.example.com:6643
  2. [student@workstation ~]$ **oc whoami --show-console**

https://console-openshift-console.apps.ocp4.example.com

* 1. Open a web browser and navigate to https://console-openshift-console.apps.ocp4.example.com. Either type the URL in a web browser, or right-click and select **Open Link** from the terminal.

|  |
| --- |
| A screenshot of a login box  Description automatically generated |

1. Log in to the OpenShift web console as the admin user.
   1. Click **Red Hat Identity Management** and log in as the admin user with the redhatocp password.

|  |
| --- |
|  |

1. View the cluster health and overall status.
   1. Review the **Cluster Overview** page.

If you do not see this page after a successful login, then locate the left panel from the OpenShift web console. If you do not see the left panel, then click the main menu icon at the upper left of the web console. Navigate to **Home** → **Overview** to view general cluster information.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

The **Overview** section contains links to helpful documentation and an initial cluster configuration walkthrough.

* 1. Scroll down to view the **Status** section, which provides a summary of cluster performance and health.

A screenshot of a computer

Description automatically generated

Many of the headings are links to sections with more detailed cluster information.

* 1. Continue scrolling to view the **Cluster utilization** section, which contains metrics and graphs that show resource consumption.

A screenshot of a computer

Description automatically generated

* 1. Continue scrolling to view the **Details** section, including information such as the cluster API address, cluster ID, and Red Hat OpenShift version.

A screenshot of a computer

Description automatically generated

* 1. Scroll to the **Cluster Inventory** section, which contains links to the Nodes, Pods, StorageClasses, and PersistentVolumeClaim pages.

A white background with green and black lines

Description automatically generated with medium confidence

* 1. The last part of the page contains the **Activity** section, which lists ongoing activities and recent events for the cluster.

A screenshot of a computer

Description automatically generated

1. Use the OpenShift web console to access the terminal of a cluster node. From the terminal, determine the status of the kubelet node agent service and the CRI-O container runtime interface service.
   1. Navigate to **Compute** → **Nodes** to view the machine that provides the cluster resources.

A screenshot of a computer

Description automatically generated

**Note**

The classroom cluster runs on a single node named master01, which serves as the control and data planes for the cluster, and is intended for training purposes. A production cluster uses multiple nodes to ensure stability and to provide a highly available architecture.

* 1. Click the **master01** link to view the details of the cluster node.

A screenshot of a computer

Description automatically generated

* 1. Click the **Terminal** tab to connect to a shell on the master01 node.

A screen shot of a computer

Description automatically generated

With the interactive shell on this page, you can run commands directly on the cluster node.

* 1. Run the chroot /host command to enable host binaries on the node.

|  |
| --- |
|  |

* 1. View the status of the kubelet node agent service by running the systemctl status kubelet command.

A screenshot of a computer screen

Description automatically generated

Press **q** to exit the command and to return to the terminal prompt.

* 1. View the status of the CRI-O container runtime interface service by running the systemctl status crio command.

A screenshot of a computer program

Description automatically generated

Press **q** to exit the command and to return to the terminal prompt.

1. Inspect the cluster monitoring and alert rule configurations.
   1. From the OpenShift web console menu, navigate to **Observe** → **Alerting** to view cluster alert information.

A screenshot of a phone

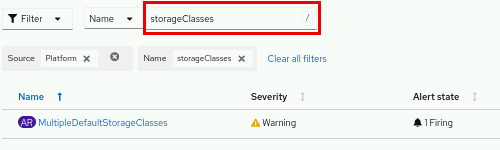
Description automatically generated

* 1. Select the **Alerting rules** tab to view the various alert definitions.

A screenshot of a computer

Description automatically generated

* 1. Filter the alerting rules by name and search for the storageClasses term.



* 1. Select the Warning alert that is labeled MultipleDefaultStorageClasses to view the details of the alerting rule. Inspect the **Description** and **Expression** definition for the rule.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Inspect cluster metrics and execute an example query.
   1. Navigate to **Observe** → **Metrics** to open the cluster metrics utility.

A screenshot of a computer

Description automatically generated

* 1. Click **Insert example query** to populate the metrics graph with sample data.

A screenshot of a computer

Description automatically generated

* 1. From the graph, hover over any point on the timeline to view the detailed data points.

A screenshot of a computer

Description automatically generated

1. View the cluster events log from the web console.
   1. Navigate to **Home** → **Events** to open the cluster events log.

A screenshot of a computer

Description automatically generated

**Note**

The event log updates every 15 minutes and can require additional time to populate entries.

* 1. Scroll down to view a chronologically ordered stream that contains cluster events.

A screenshot of a computer

Description automatically generated

**Note**

Select an event to open the Details page of the related resource.

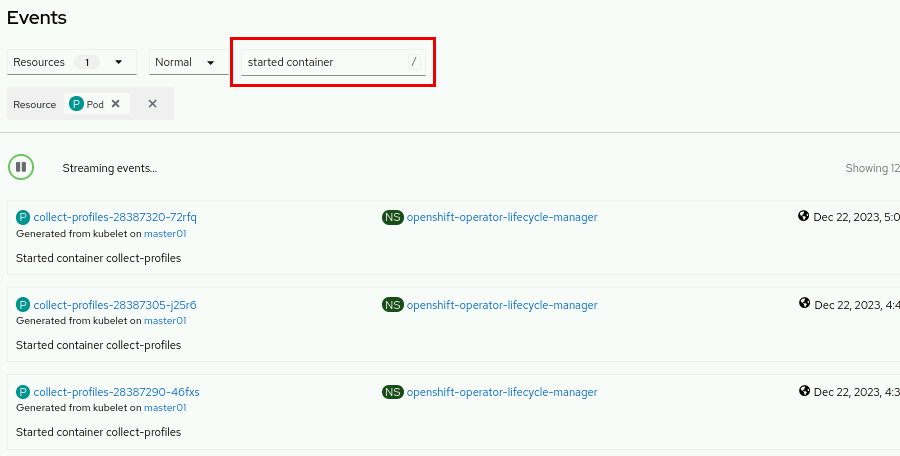
1. Filter the events by resource name, type, or message.
   1. From the **Resources** drop-down, use the search bar to filter for the pod term, and select the box labeled **Pod** to display events that relate to that resource.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

* 1. Continue to refine the filter by selecting **Normal** from the types drop-down.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

* 1. Filter the results by using the **Message** text field. Enter the started container text to retrieve the matching events.



**Finish**

On the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

[student@workstation ~]$ **lab finish intro-monitor**

[Previous](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch01s07/c2358540-87d5-48de-b49e-6f23bdcd629c)[Next](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch01s09/c2358540-87d5-48de-b49e-6f23bdcd629c)

Top of Form

|  |  |  |
| --- | --- | --- |
|  | | |
| **1.** | What is the installed version for the OpenShift cluster? |  |
| A |  | 3.9.1 |
| B |  | 4.3.2 |
| C |  | 4.14.0 |
| D |  | 5.4.3 |

1. CheckResetShow Solution

Bottom of Form

Top of Form

|  |  |  |
| --- | --- | --- |
|  | | |
| **2.** | Which three severity types are available for the alerts in the cluster? (Choose three.) |  |
| A |  | Warning |
| B |  | Firing |
| C |  | Info |
| D |  | Urgent |
| E |  | Critical |
| F |  | Oops |

1. CheckResetShow Solution

Bottom of Form

Top of Form

|  |  |  |
| --- | --- | --- |
|  | | |
| **3.** | Which three labels are on the thanos-querier route in the openshift-monitoring namespace? (Choose three.) |  |
| A |  | app.kubernetes.io/component=query-layer |
| B |  | app.kubernetes.io/instance=thanos-querier |
| C |  | app.kubernetes.io/part-of=openshift-storage |
| D |  | app.kubernetes.io/version=0.30.2 |

1. CheckResetShow Solution

Bottom of Form

Top of Form

|  |  |  |
| --- | --- | --- |
|  | | |
| **4.** | Which two objects are listed as the StorageClasses objects for the cluster? (Choose two.) |  |
| A |  | ceph-storage |
| B |  | nfs-storage |
| C |  | k8s-lvm-vg1 |
| D |  | local-volume |
| E |  | lvms-vg1 |

1. CheckResetShow Solution

Bottom of Form

Top of Form

|  |  |  |
| --- | --- | --- |
|  | | |
| **5.** | When **All Projects** is selected at the top, which three operators are installed in the cluster? (Choose three.) |  |
| A |  | MongoDB Operator |
| B |  | MetalLB Operator |
| C |  | Red Hat Fuse |
| D |  | LVM Storage |
| E |  | Package Server |

1. CheckResetShow Solution

Bottom of Form

[Previous](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch01s08/c2358540-87d5-48de-b49e-6f23bdcd629c)[Next](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch01s10/c2358540-87d5-48de-b49e-6f23bdcd629c)

Lab: Introduction to Kubernetes and OpenShift

Find essential information about your OpenShift cluster by navigating its web console.

**Outcomes**

Navigate the Red Hat OpenShift Container Platform web console to find various information items and configuration details.

As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command ensures that the Red Hat OpenShift Container Platform is deployed and ready for the lab.

[student@workstation ~]$ **lab start intro-review**

**Instructions**

1. Log in to the Red Hat OpenShift Container Platform web console, with Red Hat Identity Management as the admin user with the redhatocp password, and review the answers for the preceding quiz.
   1. Use a browser to view the login page at the https://console-openshift-console.apps.ocp4.example.com address.

|  |
| --- |
| A screenshot of a login box  Description automatically generated |

* 1. Click Red Hat Identity Management, and supply the admin username and the redhatocp password, and then click Log in to access the home page.

|  |
| --- |
| A screenshot of a login screen  Description automatically generated |

1. Hide Solution
2. View the cluster version on the Overview page for the cluster.
   1. From the **Home** → **Overview** page, scroll down to view the cluster details.

A screenshot of a computer

Description automatically generated

* 1. Locate the OpenShift version in the Details section.

|  |
| --- |
| A screenshot of a computer  Description automatically generated |

1. Hide Solution
2. View the available alert severity types within the filters on the Alerting page.
   1. Navigate to the **Observe** → **Alerting** page.

A screenshot of a computer

Description automatically generated

* 1. Click the Filter drop-down to view the available severity options.

A screenshot of a computer

Description automatically generated

1. Hide Solution
2. View the labels for the thanos-querier route.
   1. Navigate to the **Networking** → **Routes** page.

A screenshot of a computer

Description automatically generated

* 1. Type the thanos keyword in the text search field.

A screenshot of a computer

Description automatically generated

* 1. Select the thanos-querier route in the Name column.

A screenshot of a computer

Description automatically generated

* 1. Scroll down on the thanos-querier route details page to view the labels.

A screenshot of a computer

Description automatically generated

1. Hide Solution
2. View the available storage classes in the cluster.
   1. Navigate to the **Storage** → **StorageClasses** page.

A screenshot of a computer

Description automatically generated

* 1. View the available storage classes in the cluster.

A screenshot of a computer

Description automatically generated

1. Hide Solution
2. View the installed operators for the cluster.
   1. Navigate to the **Operators** → **Installed Operators** page.

A screenshot of a computer

Description automatically generated

* 1. View the list of installed operators in the cluster.

A screenshot of a computer

Description automatically generated

1. Hide Solution

**Finish**

As the student user on the workstation machine, use the lab command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

[student@workstation ~]$ **lab finish intro-review**

Summary

* A *container* is an encapsulated process that includes the required runtime dependencies for an application to run.
* Containerization addresses the application development challenges around code portability, to aid in consistently running an application from diverse environments.
* Containerization also aims to modularize applications to improve development and maintenance on the various components of the application.
* When running containers at scale, it becomes challenging to configure and deliver high availability applications and to set up networking without a container platform, such as Kubernetes.
* Pods are the smallest organizational unit for a containerized application in a Kubernetes cluster.
* Red Hat OpenShift Container Platform (RHOCP) adds enterprise-class functions to the Kubernetes container platform to deliver the wider business needs.
* Most administrative tasks that cluster administrators and developers perform are available through the RHOCP web console.
* Logs, metrics, alerts, terminal connections to the nodes and pods in the cluster, and many other features are available through the RHOCP web console.