StackRox

Definitive Guide to Red Hat OpenShift Security

Red Hat OpenShift

Definitive Guide to Red Hat OpenShift Security

OpenShift Container Platform

Red Hat's OpenShift Container Platform (OCP) is a Kubernetes platform for operationalizing container workloads remotely or as a hosted service. OpenShift enables consistent security, built-in monitoring, centralized policy management, and compatibility with Kubernetes workloads. The rapid adoption of open source projects can introduce vulnerabilities in standard Kubernetes Environments. OCP supports these projects internally, allowing users to gain open source advantages with a managed product's stability and security. OpenShift offerings include five managed and two hosted options.

This guide will focus on the Red Hat OpenShift Container Platform (RHCOP) version 4.5, which is designed to be self-managed within your infrastructure environment due to various deployment options.

OpenShift Architecture

OpenShift is built on top of Kubernetes, and while Kubernetes provides container orchestration capabilities, pod resiliency, services definitions, and deployment constructs, there are many other components required to make it work. For example, Kubernetes does not provide default Container Network Interface (CNI) or default monitoring implementations. It is up to the cluster administrator to bring additional tools to operate and manage the Kubernetes cluster and any applications running. For security teams, this presents new challenges - as an example, these teams need to create new policies and vet images, configurations, and account access for any new applications that will be deployed into the cluster.

These additional, necessary operational capabilities are provided out of the box with OCP and are pluggable so that administrators can customize components and services to meet their infrastructure needs.

OCP's architecture requires three different types of nodes within each cluster to ensure highly available deployments.

Control Plane Nodes: These nodes run the core Kubernetes control plane functions and provide additional services such as a self-service web console and developer- and operations-focused dashboards.

In most cloud environments, the control plane nodes are hidden from end-users and managed by providers for high availability, regular upgrades, and added security updates. With OCP, administrators manage, view, and interact with the control plane nodes directly, which means that they will need to set up their clusters for high availability and adequate security. To be compliant with industry-standard best practices, a minimum of three control-plane nodes should be configured to allow for accessibility to the control plane in a node outage event.

Infrastructure Nodes: These are nodes dedicated to hosting additional functionality such as OpenShift Routes and the OpenShift internal registry. Infrastructure nodes host administrator and network-focused services that are managed separately from your containerized applications.

App Nodes or Nodes: These are the OCP nodes used to run your containerized applications. These are similar to Kubernetes worker nodes and run various monitoring and networking services required across a cluster.

Cluster Design

Cloud IAM, Accounts and Limits

When using a cloud provider, you will want to enforce tight control of individual clusters and other cloud resources sharing a project. Limit access to resources by applying the principle of least privilege. Understand each provider's account roles and limitations before setting up access to any OCP clusters.

OCP helps this process by providing in-depth documentation on the installation process, including installation in AWS, Azure, GCP, and IBM Z.

Private Clusters

Strict network isolation, which prevents unauthorized external ingress to OpenShift cluster API endpoints, nodes, or pod containers, comprises a critical piece of cluster security. By default, OpenShift clusters have Kubernetes cluster API endpoints and nodes with public IP addresses. By default, the OpenShift Container Platform is provisioned using publicly-accessible DNS and endpoints. The DNS, Ingress Controller, and API server can be set to private after installing the cluster. Additionally, OpenShift may expose operations-focused dashboards for the admins and developers. Ideally, these dashboards will be running on infrastructure nodes away from your high-priority workloads.

The private cluster options vary based on the infrastructure environment. However, there are in-depth guides for setting up a private cluster through various providers. OpenShift outlines the installation methods and network setup options that are currently supported here.

After creating your private cluster, you may need to perform extra configuration steps to ensure your cluster's components are correctly set. Also, upgrades to the cluster may require Internet access and extra considerations.

Setting up a Bastion Host

A bastion host provides access to a private network from an external network and is a simple way to add an extra layer of security to your OpenShift cluster. A bastion host minimizes the chances of unauthorized access to your OCP cluster by allowing for more tightly tuned access. Benefits of a bastion host include:

- Separate login accounts for everyone accessing the bastion host
- Auditing of user access and time
- Specific node access

A bastion host is a useful way to augment security to your cluster. Restricting access to only specific nodes through the cluster using the bastions .ssh/config allows for private network access and can restrict users from tampering with nodes deemed off-limits.

Note: When using a cloud provider for deployment, utilize software-defined networks that are available. The proper implementation of cloud IAM accounts, firewall rules, and private networking will significantly reduce the attack surface.

VPC Networks

When deploying your OpenShift cluster, you will want to take advantage of the various cloud providers' built-in networking and security protections. This will vary depending on the environment; however, there are defaults and best practices to keep in mind during setup.

1. Create a single VPC network for each cluster and allow access accordingly.
2. Setup firewall rules to allow for only the ports required. These include:
    a. AWS restricted network setup
    b. Azure private network setup
    c. GCP restricted network setups

Securing etcd

By default, data stored in etcd is not encrypted at rest in the OpenShift Container Platform. Etcd encryption can be enabled in the cluster to effectively provide an additional layer of data security and canto debug in your cluster to help protect the loss of sensitive data if an etcd backup is exposed to incorrect parties. Since OpenShift recommends an etcd backup during any upgrade, encrypting etcd should be a standard practice in your organzation.

When you enable etcd encryption, the following server resources are encrypted:

- Secrets
- ConfigMaps
- Routes
- OAuth access tokens
- OAuth authorized tokens

When etcd encryption is enabled, encryption keys are created. These keys are rotated every week, and the admin must have these keys to restore from an etcd backup.

Node Images

Compromised nodes create a danger to your entire cluster and its workloads. Using minimal base operating system (OS) images and configuring read-only file systems provides two critical ways to protect your nodes against many attacks and limit their potential blast radius. With minimal images, attackers have limited tools to leverage, and if they cannot write or overwrite configuration files and binaries on the node's root file system, they cannot hijack the system as easily nor install their malicious tools.

Providers are increasingly making available minimal, container-optimized OS images such as AWS Bottlerocket and GCP's Container-Optimized OS (COS). However, It is best to leverage OpenShift's relationship with the cloud providers and use the most recent Red Hat Enterprise Linux CoreOS (RHCOS) for all of your OCP cluster's nodes. RHCOS is the default operating system for all cluster machines; however, you can create worker machines that use RHEL as their operating system.

RHCOS is designed to be as immutable as possible, allowing for only a few system settings to be changed. These settings are configured remotely, with the help of a specific operator developed by OpenShift. This scenario means no user will need to access a node directly, and any changes to the node will need to be directly authorized through the use of the Red Hat Machine Operator.

CRI-O

RHCOS also leverages CRI-O as its default container runtime. CRI-O focuses only on features needed by Kubernetes platforms. It also provides a smaller footprint and reduced attack surface than is possible with container engines that include a superset of functionality beyond Kubernetes-centric features. Since OCP is based on Kubernetes, it benefits from these features as well. By not including extra features for direct command-line use or other orchestration facilities, CRI-O's footprint is smaller, and therefore potential vulnerabilities are reduced.

Authentication and Authorization

Control Access to Cluster Resources

In addition to utilizing cloud provider IAM roles and authorization, the OCP control plane includes a built-in OAuth server. This server allows administrators to secure API access control via authentication and authorization regardless of the cluster's deployment location. OAuth 2.0 is the industry-standard protocol for authorization, which works over HTTPS and authorizes devices, servers, etc. with access tokens rather than credentials.

As an administrator, OAuth can be configured to authenticate using an identity provider, such as LDAP, GitHub, or Google. Administrators can obtain OAuth access tokens to authenticate themselves to the API as well. This feature can be enabled by cluster creation or after creation.

Manage API Access

Applications can have multiple, independent API services with different endpoints. Always aim to restrict access to endpoints and services and only grant the minimal access required. In addition to the OAuth server, OCP includes a containerized version of its 3scale API gateway. RedHat outlines this service efficiently:

"3scale gives you a variety of standard options for API authentication and security, which can be used alone or in combination to issue credentials and control access: standard API keys, application ID and key pair, and OAuth 2.0."

Utilizing the 3scale API gateway allows for fine-grained control over authorization and exposure for the cluster's API services. For example, quotas and throttling can minimize Denial-of-Service (DoS) attacks.

Rotate Cluster Certificates

Kubernetes and OpenShift clusters rely on several secure certificate chains and credentials for security. If sensitive keys or certificates are compromised, the integrity and safety of the entire cluster and its workloads may be placed at risk. Additionally, many security policies and compliance certifications require regular rotation of encryptions keys and credentials.

OCP leverages REST-based HTTPS communication with encryption via TLS certificates. These certificates are configured during installation for the components that require HTTPS traffic:

- API server and controllers
- Etcd
- Nodes
- Registry
- Router

OCP manages these certificates for the administrators and enables more control, such as allowing administrators to rotate certificates manually.

Remove the `kube-admin` Account

By default, a service account is mounted to every pod in an OCP cluster, allowing containers to send requests to the Kubernetes API server. An attacker who gains access to a pod can obtain the corresponding service account token. With RBAC enabled by default in an OCP cluster service account, privileges are determined by role bindings. If these grant elevated privileges, an attacker could send a request to the Kubernetes API server to compromise cluster resources.

Organizations can mitigate this threat vector by configuring Kubernetes RBAC and adopting the least privilege model for service accounts and their role bindings. A core example of this model is removing the kube-admin user, with OpenShift itself recommending its removal to improve cluster security.

Networking

The concept of zero-trust security has emerged to address the new security challenges of cloud-native architecture. These challenges include:

- The sharing of cloud infrastructure among workloads with different levels of trust
- Smaller microservices increasing complexity and enlarging the attack surface of applications

Microservice architecture creates a more extensive network attack surface. To address this issue, administrators and developers will have to ensure both external networks and internal software-defined networks are securely configured.

Securing Service Load Balancers

OpenShift, at a minimum, requires two load balancers: one to load balance the control plane (the control plane API endpoints) and one for the data plane (the application routers). If a load balancer is created using a cloud provider, the load balancer will be Internet-facing and may have no firewall restrictions. In most on-premises deployments, appliance-based load balancers (such as F5 or Netscaler) are used. Both types of load balancers will need to be configured by the administrator.

If the load balancer needs to be Internet-facing but should not be open to all IP addresses, you can add the field `loadBalancerSourceRanges` to the service specification to limit the IP address blocks allowed to connect. Verify that your load balancer supports this functionality. AWS, GCP, and Azure all support IP blocks.

Enable Network Policy

By default, network traffic in an OpenShift cluster is allowed between pods and can leave the cluster network altogether. Creating restrictions to allow only necessary service-to-service and cluster ingress and egress connections decrease the number of potential targets for malicious or misconfigured pods and limit their ability to exploit the cluster resources.

The OpenShift Software Defined Network (OpenShift SDN) can control network traffic to and from the cluster's pods by implementing the standard Kubernetes Network Policy API. Network Policies can control both ingress traffic and block or allow individual IP blocks. NetworkPolicy objects are additive, which means you can combine multiple NetworkPolicy objects to satisfy complex network requirements. Other Container Network Interface (CNI) implementations allow for egress rules also to be set. OpenShift SDN does not currently support that functionality.

All supported versions of OpenShift come with Network Policies enabled by default. However, the cluster while still allowing for all pod traffic to be accepted. Make sure to deny all traffic by default and create additive rules to limit pod traffic only to what is required. Test the policies to make sure they block unwanted traffic while allowing required traffic.

Master Authorized Networks

To protect against future vulnerabilities in the OpenShift API server and Kubernetes API server, limit network access to API endpoints to trusted IP addresses. Regardless of the OCP clusters, administrators need to create rules for access to the cluster's API endpoints.

Container Images

Build Secure Images

Following a few best practices for building secure container images will minimize running containers' exploitability and simplify security updates. Images containing only the files required for the application's runtime make it much more difficult for malicious attacks to compromise or exploit the containers in your cluster. Avoid using container images with frequently exploited or vulnerable tools like Linux package managers, web or other network clients, or Unix shells.

What to do:

1. Use a minimal, secure base image.
2. Only install tools needed for the container's application. Debugging tools should be omitted from production containers.
3. Remove exploitable tools as part of the initialization process with init-containers.
4. Remove the package manager from the image as a Dockerfile build step.
5. Keep images up-to-date. This practice includes watching for new versions of both the base image and any third-party tools you install.s
6. Maintain compatibility with proper tags. Avoid the use of the "latest" tag to help downstream users avoid compatibility issues.

Manage Images with a Registry

OpenShift offers an integrated image registry located on the infrastructure nodes of the cluster. This setup allows an organization to avoid third-party hosting and public image storing services, such as Docker Hub. By keeping all necessary images within the cluster, organizations can avoid dependency on a third-party service and any outages associated with them.

Organizations can enact more tuned policies for image use and vetting by taking advantage of a local image repository. Only after an image has passed the proper controls should it be allowed to be used to deploy containers into your cluster.

Use Verified Images from the Container Catalog

To help select secure base images, Red Hat offers a container image catalog. Images in the catalog are maintained by Red Hat, meaning they are frequently scanned and updated. The container catalog also provides an image's history; every time it is updated, a new tag is created. According to a simple scale A to F, the image is graded based on creation date and known security flaws.

Use an Image Scanner

Using containers free of known software security vulnerabilities requires ongoing vigilance. All the images deployed to a cluster should be scanned regularly by an image scanner that keeps an up-to-date database of CVEs (Common Vulnerabilities and Exposure). Several open-source and proprietary options exist, but be sure to choose a solution that scans for vulnerabilities in operating system packages and in third-party runtime libraries for the programming languages your applications use.

To address CVEs when they are found in your internally maintained images, your organization should have policies for updating and replacing images known to have serious, fixable vulnerabilities for images that are already deployed. Image scanning should be part of your CI/CD pipeline process, and images with high-severity, fixable CVEs should generate an alert and fail a build.

OpenShift Container Security Operator

OpenShift allows for a Container Security Operator (CSO) to be installed in the cluster. This Operator will view results from the OpenShift Container Platform web console for container images used to launch running pods in the cluster. The CSO can be directed to look at all pods or specific namespaces. It will then query the container registry where the image came from for any vulnerabilities. The Operator will also access vulnerabilities based on how long the container has been used without an upgrade or patch.

The CSO's findings can be viewed through the OCP dashboard that outlines the cluster's overall health and specific images that require attention.

Runtime Security for Workloads

Projects and Namespaces

Kubernetes namespaces provide scoping for cluster objects, allowing fine-grained cluster object management. Kubernetes Role-based Access Control (RBAC) rules for most resource types apply at the namespace level. Controls like network policies and many add-on tools and frameworks like service meshes also often scoped to the namespace level.

OpenShift expands on the Kubernetes namespace functionality with OpenShift projects. A project is simply a Kubernetes namespace with additional annotations, and just like namespaces, users must be granted access to create and use project resources.

Projects allow the user to add more metadata to a Kubernetes namespace and provide more context in the OCP web console. Projects can have a separate name, displayName, and description, with displayName and description being option inputs to give a more specific name and description in the web console.

When creating OpenShift projects, it is best to plan out how you will separate the projects in the cluster. Since RBAC and network policies integrate with the Kubernetes Namespaces, it is essential to evaluate how their implementation will affect your applications. To start, configuring one namespace per application provides the best opportunity to implement excellent security protocols. The fine-grained control using network policies and RBAC will add slightly more management and more secure scalability. Also, avoid using the default namespace in any cluster outside of a development cluster. All applications require their namespace and should not be deployed into default due to ease of deployment.

Kubernetes RBAC

Kubernetes Role-Based Access Control provides the standard method for managing authorization for the Kubernetes API endpoints. The practice of creating and managing comprehensive RBAC roles that follow the principle of least privilege and provide some of the most critical protections possible for your OPC clusters. By utilizing a principle of least privilege and regular audits, we can limit bad actors, internal misconfigurations, and accidents.

RBAC is enabled by default in OpenShift clusters and includes the same set of default cluster roles that can be found in Kubernetes clusters. Configuring RBAC effectively and securely requires some understanding of the Kubernetes API. You can start with the official documentation, read about some best practices, and you can also find an in-depth explanation from the OpenShift documentation.

When working with OCP, we want to create all necessary RBAC resource objects for the cluster workloads and test them in a non-production environment. Once your team has a solid working knowledge of RBAC, create some internal policies and guidelines. Make sure you also regularly audit your Role permissions and RoleBindings. Pay special attention to minimizing the use of ClusterRoles and ClusterRoleBindings, as these apply globally across all namespaces and to resources that do not support namespaces. (You can use the output of kubectl api-resources in your cluster to see which resources are not namespace-scoped.)

Limit Container Runtime Privileges

Most containerized applications will not need any special host privileges on the node to function correctly. By following the principle of least privilege and minimizing your cluster's running containers' capabilities, you can significantly reduce the level of exploitation for malicious containers and of accidental damage by misbehaving applications. With Kubernetes, the PodSpec Security Context is used to define the exact runtime requirements for each workload. With RHOCP, Security Context Constraints (SCCs) are used to restrict privileges for pods. Similar to how RBAC resources control user access, administrators can use SCCs to control permissions for pods.

SCCs are OpenShift resources; they define a set of conditions (or rules) that a pod must satisfy to be created (or admitted in the cluster). These controls can limit the resources, system calls, and filesystem access of the pods running in the cluster. Using SCCs, the level of privileges are controlled for the application, and if needed, give them more permissive or more restrictive privileges.

Red Hat outlines the capabilities of SCCs in their documentation:

- Whether a pod can run privileged containers.
- The capabilities that a container can request.
- The use of host directories as volumes.
- The SELinux context of the container.
- The container user ID.
- The use of host namespaces and networking.
- The allocation of an FSGroup that owns the pod's volumes.
- The configuration of allowable supplemental groups.
- Whether a container requires the use of a read-only root file system.
- The usage of volume types.
- The configuration of allowable seccomp profiles.

By default, for authenticated users, resources deployed in a project inherit a default security context associated with the authenticated users role. An OpenShift cluster contains eight default SCC's that can be applied to authenticated users:

- anyuid
- hostaccess
- hostmount-anyuid
- Hostnetwork
- node-exporter
- non-root
- privileged
- restricted

Make sure not to tamper with these default SCCs since they are used for essential cluster functions. Instead, create new SCCs for specific users and limit their capabilities.

Some guidelines when creating new SCCs include:

1. Do not allow containers to run as root. Running as root creates the most significant risk since root access in a container is equal to root access on the underlying node.
2. Do not use the host network or process space. Again, these settings create the potential for compromising the node and every container running on it.
3. Do not allow privilege escalation.
4. Use a read-only root filesystem in the container.
5. Use the default (masked) /proc filesystem mount.
6. Drop unused Linux capabilities and do not add optional capabilities that your application does not require. (Available capabilities depend on the container runtime in use on the nodes. GKE nodes can use either Docker or containerd, depending on the node image.)
7. Use SELinux options for more fine-grained process controls.
8. Give each application its own Kubernetes Service Account rather than sharing or using the namespace's default service account.
9. Do not mount the service account token in a container if it does not need to access the Kubernetes API.

Monitoring & Maintenance

Cluster Upgrades

Support Policy

Keeping up-to-date with patches for your cluster's OpenShift version is an important requirement for your cluster and workload security. OpenShift provides full support of the newest released versions until a month after the newest version is released. OpenShift will support the maintenance of the three most recent minor releases.

Release Channels

OpenShift Container Platform 4.5 offers the following upgrade channels:

- Candidate-4.5
- Fast-4.5
- Stable-4.5

The candidate-4.5 channel contains candidate builds for a z-stream (4.5.z) release. Release candidates contain all the features of the product but are not supported. Use release candidate versions to test feature acceptance and assist in qualifying the next OPC version. The fast-4.5 channel is updated with new 4.5 versions as soon as Red Hat declares the given version as a general availability release. These releases are fully supported, production quality, and have performed well while available as a release candidate in the candidate-4.5 channel from where they were promoted.

While the fast-4.5 channel contains releases as soon as their errata are published, releases are added to the stable-4.5 channel after a delay. Data is collected from Red Hat SRE teams, Red Hat support services, and pre-production and production environments that participate in a connected customer program about the release's stability during this delay. You can use the stable-4.5 channel to upgrade from a previous minor version of the OpenShift Container Platform.

Upgrade Recommendations

OpenShift allows for multiple ways of upgrading a cluster. A minor upgrade can be made through the web console or through the CLI. Updating a restricted cluster will need to be done via the command line and will require administrative control and Internet access to pull the most recent OpenShift images.

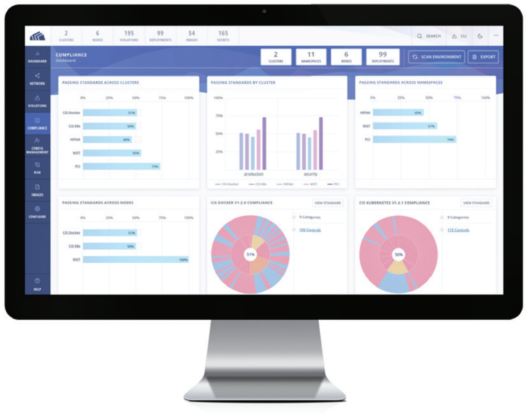
Regardless of the method chosen, make sure to have a recent etcd backup in case of a failed upgrade. Also, ensure all Operators are upgraded to the latest version on their channel. This will ensure that the Operators will be valid for the upgraded cluster.

Audit Logging

Logging events and changes at both the OpenShift level and the node level creates a critical audit trail to use for evaluating your cluster's security, especially in case of a breach or attack. Audit logs are available by default and can be easily accessed through the command line. This is vital for restricted clusters that are not using the web interface to view cluster logs.

OpenShift allows administrators to view audit logs from the node and API directly from the command line. Use the OpenShift uses Elasticsearch and Fluentd for logging and storing your cluster events. Make sure to take advantage of these managed open source tools to debug in your cluster effectively.

StackRox



Ready to see StackRox in action?

Get a personalized demo tailored for your business, environment, and needs.

REQUEST DEMO