A layered approach to container and Kubernetes security

Securing containers from build to deploy to run

Table of contents

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

Introduction

Containers have garnered broad appeal through their ability to package an application and its dependencies into a single image that can be promoted from development, to test, and to production. Containers make it easy to ensure consistency across environments and across multiple deployment targets like physical servers, virtual machines (VMs), and private or public clouds. With containers, teams can more easily develop and manage the applications that deliver business agility.

- **Applications:** Containers make it easier for developers to build and promote an application and its dependencies as a unit. Containers can be deployed in seconds. In a containerized environment, the software build process is the stage in the life cycle where application code is integrated with needed runtime libraries.
- **Infrastructure:** Containers represent sandboxed application processes on a shared Linux® OS kernel. They are more compact, lighter, and less complex than virtual machines and are portable across different environments—from on-premises to public cloud platforms.

Kubernetes is the container orchestration platform of choice for the enterprise. With many organizations now running essential services on containers, ensuring container security has never been more critical. This paper describes the key elements of security for containerized applications.

Comprehensive container and Kubernetes security: Layers and life cycle

Securing containers is a lot like securing any running Linux process. You need to think about security throughout the layers of the solution stack before you deploy and run your container. You also need to think about security throughout the application and container life cycle. Importantly, security must be a continuous process that is integrated throughout the IT life cycle as well, extending to respond to new threats and solutions as they emerge. Figure 1 illustrates a comprehensive approach to container security.

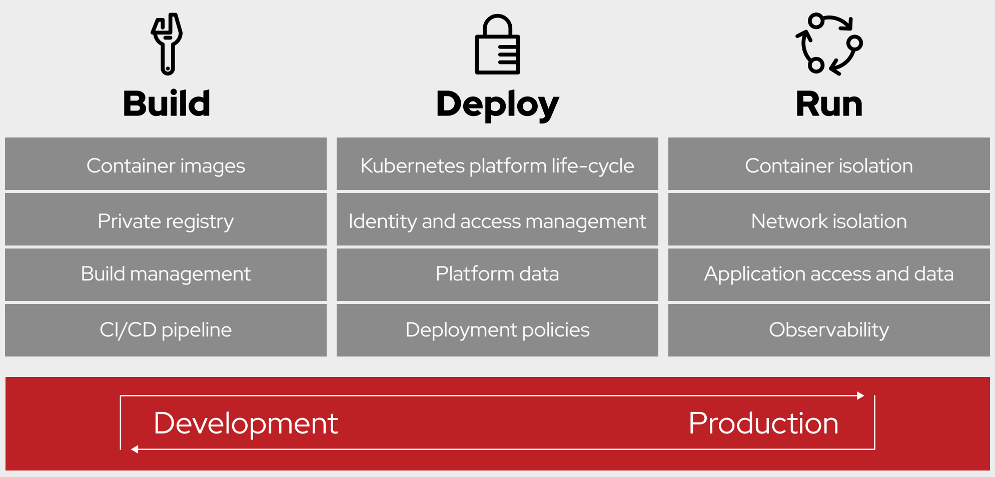

Figure 1. Comprehensive container security involves securing the build, deployment, and runtime, and extending security with specialized tools when appropriate.

Containers make it easier for developers to build and promote an application and its dependencies as a unit. Containers also make it easy to get the most use of your servers by allowing for multitenant application deployments on a shared host. You can easily deploy multiple applications on a single host, spinning up and shutting down individual containers as needed. Unlike traditional virtualization, you do not need a hypervisor to manage guest operating systems on each VM. Containers virtualize your application processes, not your hardware.

Of course, applications are rarely delivered in a single container. Even simple applications typically have a frontend, a backend, and a database. And deploying modern microservices-based applications in containers means deploying multiple containers—sometimes on the same host and sometimes distributed across multiple hosts or nodes as shown in Figure 2.

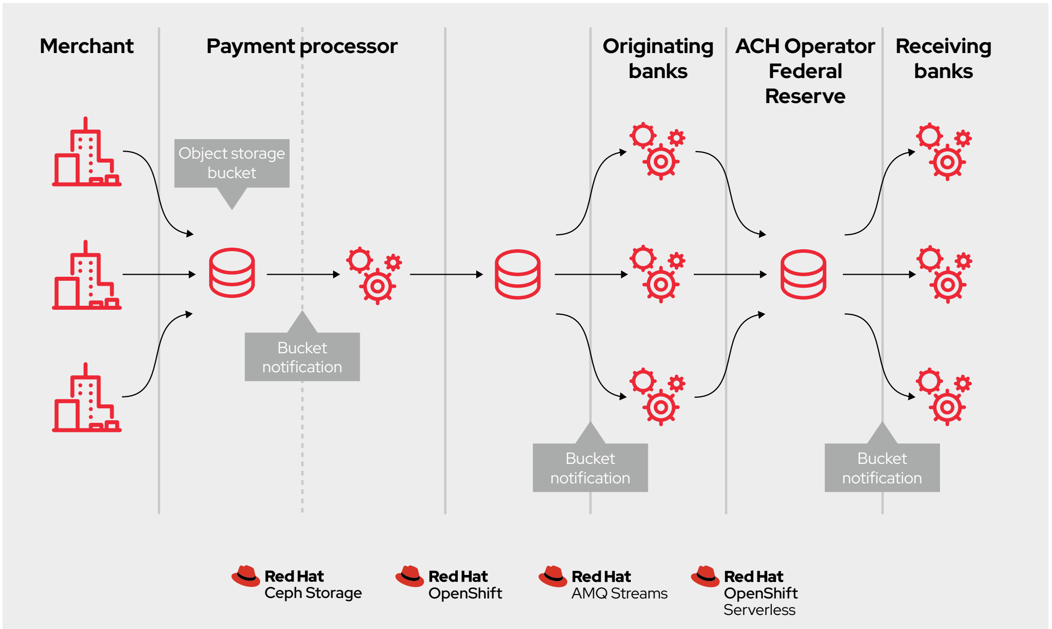

Figure 2. Container environments can be complex and distributed across multiple nodes.

When managing container deployment at scale, you need to consider:

- Which containers should be deployed to which hosts?
- Which host has more capacity?
- Which containers need access to each other and how will they discover each other?
- How do you control access to and management of shared resources such as network and storage?
- How do you monitor container health?
- How do you automatically scale application capacity to meet demand?
- How do you enable developer self-service while also meeting security requirements?

You can build your own container management environment, which requires spending time integrating and managing individual components. Or you can deploy a container platform with built-in management and security features. This approach lets your team focus their energies on building the applications that provide business value rather than reinventing infrastructure.

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

Build security into your applications

Building security into your applications is critical for cloud-native deployments. Securing your containerized applications requires that you:

1. Use trusted container content.
2. Use an enterprise container registry.
3. Control and automate building containers.
4. Integrate security into the application pipeline.

1. Use trusted container content

When managing security, what is inside your container matters. For some time now, applications and infrastructures have been composed from readily available components. Many of these are open source packages, such as the Linux operating system, Apache Web Server, Red Hat JBoss® Enterprise Application Platform, PostgreSQL, and Node.js. Containerized versions of these packages are also available so you can avoid building your own. However, as with any code you download from an external source, you need to know where the packages originated, who built them, and whether they contain any malicious code. Ask yourself:

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

Red Hat has been packaging and delivering trusted Linux content for years in Red Hat Enterprise Linux and across our portfolio. Red Hat is now delivering that same trusted content packaged as Linux containers. With the introduction of Red Hat Universal Base Images, you can take advantage of the greater reliability, security, and performance of Red Hat container images wherever Open Container Initiative (OCI)-compliant Linux containers run. This means you can build a containerized application on Red Hat Universal Base Image, push it to the container registry of your choice, and share it.

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

Red Hat continuously monitors the health of the images it delivers. The Container Health Index exposes the "grade" of each container image, detailing how container images should be curated, consumed, and evaluated to meet the needs of production systems. Containers are graded based in part on the age and impact of unapplied security errata to all components of a container, providing an aggregate rating of container safety that can be understood by security experts and nonexperts alike.

When Red Hat releases security updates—such as fixes to runc CVE-2019-5736, MDS CVE-2019-11091, or VHOST-NET CVE-2019-14835—we also rebuild our container images and push them to the public registry. Red Hat security advisories alert you to any newly discovered issues in certified container images and direct you to the updated image so that you can, in turn, update any applications that use the image.

There may be times when you need content that Red Hat does not provide. We recommend using container scanning tools that use continuously updated vulnerability databases to be sure you always have the latest information on known vulnerabilities when using container images from other sources. Because the list of known vulnerabilities is constantly evolving, you need to check the contents of your container images when you first download them and continue to track vulnerability status over time for all your approved and deployed images, just as Red Hat does for Red Hat container images.

2. Use an enterprise container registry for more secure access to container images

Of course, your teams are building containers that layer content on top of the public container images you download. You need to manage access to, and promotion of, the downloaded container images and the internally built images the same way you manage other types of binaries. There are a number of private registries that support storage of container images. We recommend selecting a private registry that helps you automate policies for the use of container images stored in the registry.

Red Hat OpenShift includes a private registry that provides basic functionality to manage your container images. The Red Hat OpenShift registry provides role-based access control (RBAC) that allows you to manage who can pull and push specific container images. Red Hat OpenShift also supports integration with other private registries you may already be using, such as JFrog's Artifactory and Sonatype Nexus.

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

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

3. Control and automate building container images

Managing this build process is key to securing the software stack. Adhering to a "build once, deploy everywhere" philosophy ensures that the product of the build process is exactly what is deployed in production. It is also important to maintain the immutability of your containers. In other words, do not patch running containers—rebuild and redeploy them instead.

Red Hat OpenShift provides a number of capabilities for automating builds based on external events, as a way to improve the security of your custom images.

- Red Hat Quay triggers provide a mechanism for spawning a repository build of a Dockerfile from an external event such as a GitHub push, BitBucket push, GitLab push, or webhook.
- Source-to-image (S2I) is an open source framework for combining source code and base images (Figure 3). S2I makes it easy for your development and operations teams to collaborate on a reproducible build environment. When a developer commits code with git, under S2I, Red Hat OpenShift can:
  - Trigger automatic assembly of a new image from available artifacts, including the S2I base image, and the newly committed code (via webhooks on the code repository or some other automated CI process).
  - Automatically deploy the newly built image for testing.
  - Promote the tested image to production status and automatically deploy the new image through the continuous integration and deployment (CI/CD) process.

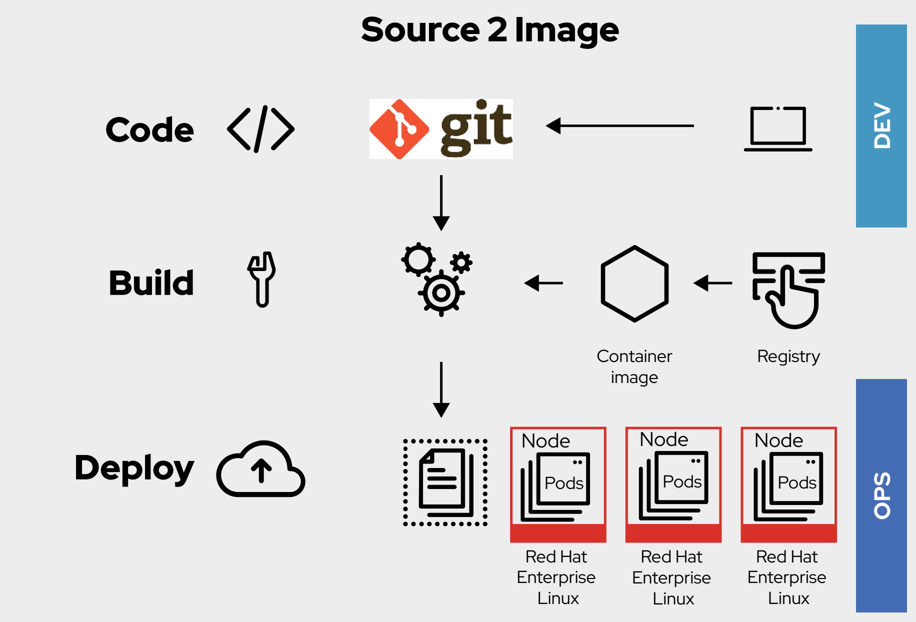

Figure 3. S2I is a toolkit and workflow for building reproducible ready-to-run container images from source code.

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

  For example, consider an application built with three container image layers: base, middleware, and the application layer. An issue is discovered in the base image and that image is rebuilt by Red Hat and pushed to Red Hat's ecosystem catalog. With image streams enabled, Red Hat OpenShift can detect that the image has changed. For builds that are dependent on this image and that have triggers defined, Red Hat OpenShift will automatically rebuild the application image, incorporating the fixed base image.

  Once the build is complete, the updated custom image is pushed to Red Hat OpenShift's internal registry. Red Hat OpenShift immediately detects changes to images in its internal registry and, for applications where triggers are defined, automatically deploys the updated image, ensuring that the code running in production is always identical to the most recently updated image. All of these capabilities work together to integrate security capabilities into your CI/CD process and pipeline.

4. Integrate security into the application pipeline

Red Hat OpenShift includes integrated instances of Jenkins for CI and Tekton, a next-generation Kubernetes CI/CD pipeline that works for containers (including serverless). Red Hat OpenShift also includes rich RESTful APIs that you can use to integrate your own build or CI/CD tools including a private image registry.

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

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

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

CI/CD: Scanners can be integrated with CI for real-time checking against known vulnerabilities that catalog the open source packages in your container, notify you of any known vulnerabilities, and update you when new vulnerabilities are discovered in previously scanned packages.

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

Finally, we recommend that you sign custom built containers so that you can be sure they are not tampered with between build and deployment.

Deploy: Managing the configuration, security, and compliance of your deployment

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

1. Platform configuration and life cycle management.
2. Identity and access management.
3. Securing platform data and attached storage.
4. Deployment policies.

5. Platform configuration and life cycle management

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

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

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

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

6. Identity and access management

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

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

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

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

7. Securing platform data

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

Red Hat OpenShift protects platform data in transit by:

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

Red Hat OpenShift protects platform data at rest by:

- Optionally encrypting Red Hat Enterprise Linux CoreOS disks and the etcd datastore for additional security.
- Providing Federal Information Processing Standards (FIPS) readiness for Red Hat OpenShift. FIPS 140-2 is a U.S. government security standard used to approve cryptographic modules. When Red Hat Enterprise Linux CoreOS is booted in FIPS mode, Red Hat OpenShift platform components call Red Hat Enterprise Linux cryptographic modules.

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

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

- A **persistent volume (PV)** can be mounted on a host in any way supported by the resource provider. Providers will have different capabilities and each PV's access modes are set to the specific modes supported by that particular volume. For example, NFS can support multiple read/write clients, but a specific NFS PV might be exported on the server as read-only. Each PV gets its own set of access modes describing that specific PV's capabilities. Such as ReadWriteOnce, ReadOnlyMany, and ReadWriteMany.
- For **shared storage** (e.g., NFS, Ceph, Gluster), the trick is to have the shared storage persistent volume (PV) register its group ID (gid) as an annotation on the PV resource. When the PV is claimed by the pod, the annotated gid will be added to the supplemental groups of the pod and give that pod access to the contents of the shared storage.
- For **block storage** (e.g., EBS, GCE Persistent Disks, iSCSI), container platforms can use SELinux capabilities to secure the root of the mounted volume for non-privileged pods, making the mounted volume owned by, and only visible to, the container it is associated with.

Of course, you should take advantage of the security features available in your chosen storage solution.

8. Automate policy-based deployment

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

- Policy-based container deployment

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

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

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

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

- Policy-based multicluster management

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

- **Multicluster life-cycle management** that allows you to create, update, and destroy Kubernetes clusters reliably, consistently, and at scale.
- **Policy-driven governance risk and compliance** utilizes policies to automatically configure and maintain consistency of security controls according to industry corporate standards. You can also specify a compliance policy to apply across one or more managed clusters.

Protect running applications

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

1. Container isolation.
2. Application and network isolation.
3. Securing application access.
4. Observability.

9. Container isolation

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

Containers are Linux processes with isolation and resource confinement that let you run sandboxed applications on a shared host kernel. Your approach to securing containers should be the same as your approach to securing any running process on Linux.

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

Every Linux container running on a Red Hat OpenShift platform is protected by powerful Red Hat Enterprise Linux security features built into Red Hat OpenShift nodes. Linux namespaces, SELinux, Cgroups, capabilities, and secure computing mode (seccomp) are used to secure containers running on Red Hat Enterprise Linux.

- Linux namespaces provide the fundamentals of container isolation. A namespace makes it appear to the processes within the namespace that they have their own instance of global resources. Namespaces provide the abstraction that gives the impression you are running on your own operating system from inside a container.
- SELinux provides an additional layer of security to keep containers isolated from each other and from the host. SELinux allows administrators to enforce mandatory access controls (MAC) for every user, application, process, and file. SELinux is like a brick wall that will stop you if you manage

to break out of the namespace abstraction (accidentally or on purpose). SELinux mitigates container runtime vulnerabilities, and well-configured SELinux configurations can prevent container processes from escaping their containment.

- Cgroups (control groups) limit, account for, and isolate the resource usage (e.g., CPU, memory, disk I/O, network) of a collection of processes. Use Cgroups to prevent your container resources from being stomped on by another container on the same host. Cgroups can also be used to control pseudo devices—a popular attack vector.
- Linux capabilities can be used to lock down privileges in a container. Capabilities are distinct units of privilege that can be independently enabled or disabled. Capabilities allow you to do things such as send raw internet protocol (IP) packets or bind to ports below 1024. When running containers, you can drop multiple capabilities without impacting the vast majority of containerized applications.
- Finally, a secure computing mode (seccomp) profile can be associated with a container to restrict available system calls.

10. Application and network isolation

Multitenant security is essential for enterprise-scale use of Kubernetes. Multitenancy allows you to have different teams use the same cluster while preventing unauthorized access to each other's environments. Red Hat OpenShift supports multitenancy through a combination of kernel namespaces, SELinux, RBAC, Kubernetes (project) namespaces, and network policies.

- **Red Hat OpenShift projects** are Kubernetes namespaces with SELinux annotations. Projects isolate applications across teams, groups, and departments. Local roles and bindings are used to control who has access to individual projects.
- **Security context constraints** let you drop privileges by default, which is important and still the best practice. Red Hat OpenShift security context constraints (SCCs) ensure that, by default, no privileged containers run on OpenShift worker nodes. Access to the host network and host process IDs are denied by default.

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

- **Ingress traffic control.** Red Hat OpenShift includes CoreDNS to provide a name resolution service to pods. The Red Hat OpenShift router (HAProxy) supports ingress and routes to provide external access to services running on-cluster. Both support reencrypt and passthrough policies: "reencrypt" decrypts and reencrypts HTTP traffic when forwarding it whereas "passthrough" passes traffic through without terminating TLS.
- **Network namespaces.** The first line in network defenses comes from network namespaces. Each collection of containers (known as a "pod") gets its own IP and port range to bind to, thereby isolating pod networks from each other on the node. The pod IP addresses are independent of the physical network that Red Hat OpenShift nodes are connected to.

- **Network policies:** The Red Hat OpenShift SDN uses network policies to provide fine-grained control of communication between pods. Network traffic can be controlled to any pod from any other pod, on specific ports and in specific directions. When network policies are configured in multitenant mode, each project gets its own virtual network ID, thereby isolating project networks from each other on the node. In multitenant mode (by default) pods within a project can communicate with each other but pods from different namespaces cannot send packets to or receive packets from pods or services of a different project.
- **Egress traffic control.** Red Hat OpenShift also provides the ability to control egress traffic from services running on the cluster using either router or firewall methods. For example, you can use IP whitelisting to provide access to an external database.

11. Securing application access

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

Controlling user access

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

Controlling API access

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

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

Securing application traffic

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

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

Red Hat OpenShift Service Mesh provides discovery, load balancing, and, key to security, service-to-service authentication and encryption, failure recovery, metrics, and monitoring.

3scale Istio Adapter is an optional adapter that allows you to label a service running within Red Hat OpenShift Service Mesh.

12. Observability

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

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

Auditing events is a security best practice and generally required to comply with regulatory frameworks. At its core, Red Hat OpenShift auditing was designed using a cloud-native approach to provide both centralization and resiliency. In Red Hat OpenShift, host auditing and event auditing are enabled by default on all nodes. Red Hat OpenShift provides extraordinary flexibility for configuring management and access to auditing data. You can control the amount of information that is logged to the API server audit logs by choosing which audit log policy profile to use.

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

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

Extending security with a robust ecosystem

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

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

Conclusion

Deploying container-based applications and microservices is not just about security. Your container platform needs to provide an experience that works for your developers and your operations team. You need a security-focused, enterprise-grade, container-based application platform that empowers developers and operators without compromising the functions needed by each team, while also improving operational efficiency and infrastructure utilization.

Red Hat OpenShift is built on a core of standard and portable Linux containers that deliver built-in security features, including:

- Integrated build and CI/CD tools for secure DevOps practices.
- Hardened, enterprise-ready Kubernetes with built-in platform configuration, compliance, and lifecycle management.
- Strong RBAC with integrations to enterprise authentication systems.
- Options for managing cluster ingress and egress.
- Integrated SDN and service mesh with support for network microsegmentation.
- Support for securing remote storage volumes.
- Red Hat Enterprise Linux CoreOS, optimized for running containers at scale with strong isolation.
- Deployment policies to automate runtime security.
- Integrated monitoring, audit, and logging.

Red Hat OpenShift also provides the largest collection of supported programming languages, frameworks, and services (Figure 4). Red Hat Advanced Cluster Management for Kubernetes provides tightly integrated multicluster management.

Red Hat OpenShift is available to run on OpenStack, VMware, bare metal, AWS, Google Cloud Platform (GCP), Azure, IBM Cloud and any platform that supports Red Hat Enterprise Linux. Red Hat also provides Red Hat OpenShift Dedicated on AWS and GCP as a public cloud service. Azure Red Hat OpenShift is jointly offered by Red Hat and Microsoft. Red Hat OpenShift Service on AWS is jointly offered by Red Hat and Amazon.

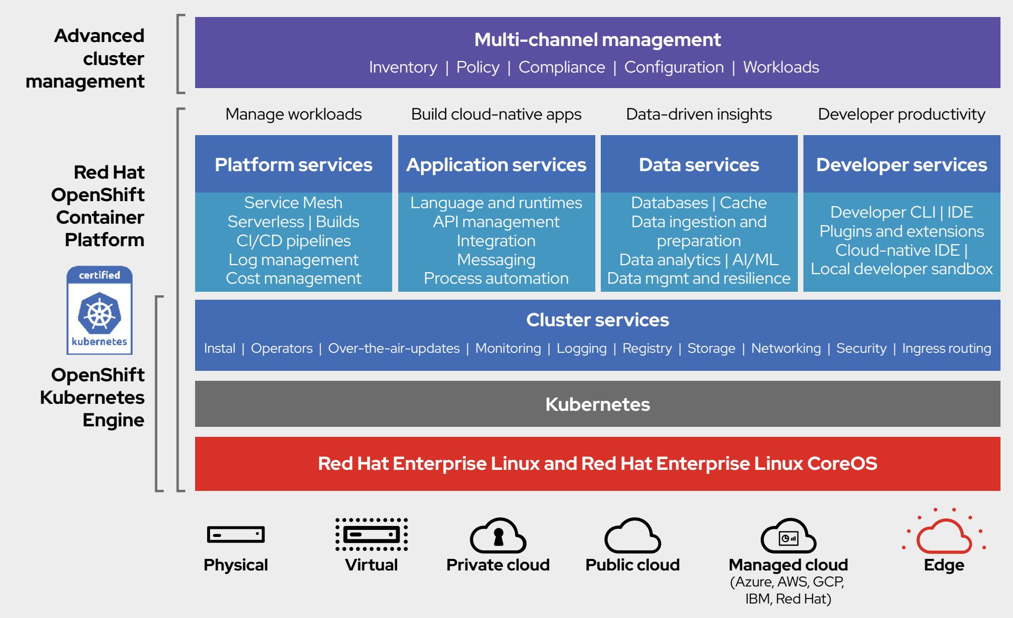

Figure 4. Red Hat OpenShift provides a full featured technology stack across physical virtual, private, and public clouds.

As a leading provider bringing trusted open source solutions to enterprise customers for over two decades, Red Hat brings this same level of trust and security to containers through solutions like Red Hat OpenShift Container Platform, Red Hat Advanced Cluster Management for Kubernetes, and our container-enabled Red Hat product portfolio.