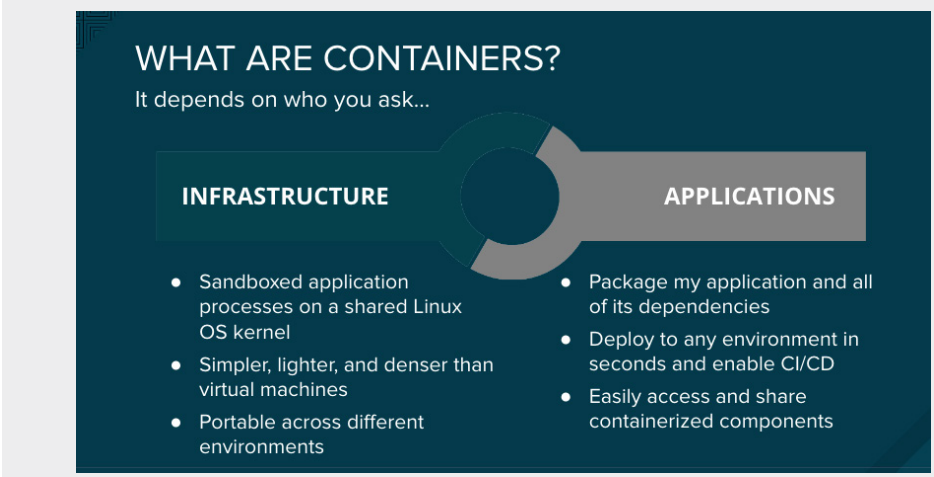
TECHNOLOGY DETAIL

TEN LAYERS OF CONTAINER SECURITY

INTRODUCTION

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



Enterprises require strong security and anyone running essential services in containers will ask, "Are containers secure?" and "Can we trust containers with our applications?" This paper describes 10 key elements of security for different layers of the container solution stack and different stages of the container life cycle.

SECURING CONTAINERS: LAYERS AND LIFE CYCLE

Securing containers is a lot like securing any running 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. The 10 key elements of container security are:

1. Container host multitenancy
2. Container content
3. Container registries
4. Building containers
5. Deploying containers
6. Container orchestration
7. Network isolation
8. Storage
9. Application programming interface (API) management
10. Federated clusters

1. The container host operating system and multitenancy

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 enabling 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. And, unlike traditional virtualization, you do not need a hypervisor or to manage guest operating systems on each VM. Containers virtualize your application processes, not your hardware.

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

Containers are Linux® processes with isolation and resource confinement that enable you to 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. Dropping privileges is important and still the best practice. Even better is to create containers with the least privilege possible. Containers should run as user, not root. Next, make use of the multiple levels of security available in Linux. Linux namespaces, Security-Enhanced Linux (SELinux), Cgroups, capabilities, and secure computing mode (seccomp) are five of the security features available for securing 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 when you are 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 (accidentally or on purpose) the namespace abstraction.
- 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 ensure your container will not be 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 root 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 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.

You can further enhance security for your applications and infrastructure by deploying your containers to a lightweight operating system optimized to run Linux containers, like Red Hat Enterprise Linux Atomic Host. Atomic Host reduces the attack surface by minimizing the host environment and tuning it for containers.

As a proof point of the security features available to you with Red Hat Enterprise Linux and Atomic Host, Red Hat Enterprise Linux 7.1 recently received Common Criteria certification, including certification of the Linux Container Framework Support.

Traditional virtualization also enables multitenancy—but in a very different way. Virtualization relies on a hypervisor initializing guest VMs, each of which has its own OS, as well as the running application and its dependencies. With VMs, the hypervisor isolates the guests from each other and from the host. Fewer individuals and processes have access to the hypervisor, reducing the attack surface on the physical server. However, security must still be monitored for threats; for example, one guest VM may be able to use hypervisor bugs to gain access to another VM or the host kernel. And, when the OS needs patching, it must be patched on all guest VMs using that OS.

Containers can be run inside guest VMs and there may be use cases where this is desirable. For example, if you are deploying a traditional application in a container, perhaps in order to lift and shift an application to the cloud, you may wish to place the container inside a guest VM. However, container multitenancy on a single host provides a more lightweight, flexible, and easier-to-scale deployment solution. This deployment model is particularly appropriate for cloud-native applications.

2. Container content (use trusted sources)

When it comes to security, what's 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 now also readily available so that you do not have to build your own. But, as with any code you download from an external source, you need to know where the packages originally came from, who built them, and whether there's any malicious code inside them. Ask yourself:

- Will what's inside the containers compromise my infrastructure?
- Are there known vulnerabilities in the application layer?
- Are the runtime and OS layers up to date?
- How frequently will the container be updated and how will I know when it's 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. This includes base images for Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux 6 and also provides a large number of certified images for various language runtimes, middleware, databases, and more via the Red Hat Container Catalog. Red Hat certified containers run anywhere Red Hat Enterprise Linux runs, from bare metal to VMs to cloud. Certified containers are supported by Red Hat and our partners.

Container image content is packaged from known source code. Red Hat also provides security monitoring. With its new Container Health Index, Red Hat 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 just how safe a container is that can be understood by security experts and nonexperts alike.

When Red Hat releases security updates—such as fixes to glibc, Drown, or Dirty Cow—we also rebuild our container images and push them to our 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.

Of course, there will 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.

Red Hat provides a pluggable API in Red Hat Enterprise Linux to support multiple scanners such as OpenSCAP, Black Duck Hub, JFrog Xray, and Twistlock. Red Hat CloudForms can also be used with OpenSCAP to scan container images for security issues. Also, Red Hat OpenShift gives you the ability to use scanners with your continuous integration and continuous delivery (CI/CD) process. This is covered in more detail below.

3. Container registries (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 in 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 can be used to manage your container images. The OpenShift registry provides role-based access controls that allow you to manage who can pull and push specific container images. OpenShift also supports integration with other private registries you may already be using, such as JFrog's Artifactory and Docker Trusted Registry.

The list of known vulnerabilities is constantly evolving, so you need to track the contents of your deployed container images, as well as newly downloaded images, over time. Your registry should include features that help you manage content based on metatdata about the container, including known vulnerabilities. For example, you can use Red Hat CloudForms SmartState analysis to flag vulnerable images in your registry. Once flagged, OpenShift will prevent that image from being run going forward.

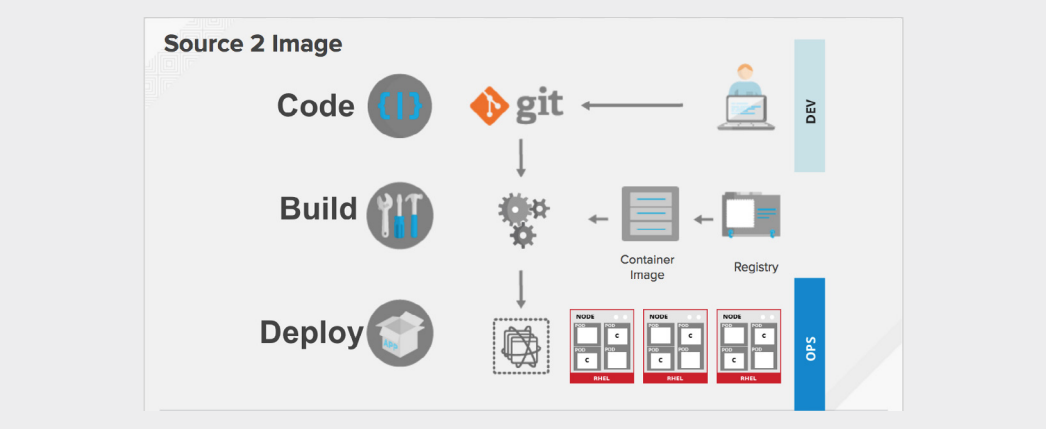
We will highlight some of the additional ways that OpenShift supports automating security policies for container content in the next few sections.

4. Security and the build process

In a containerized environment, the software build process is the stage in the life cycle where application code is integrated with needed runtime libraries. 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's 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 build management and security:

- Source-to-image (S2I) is an open source framework for combining source code and base images. 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, OpenShift:
  - Can trigger (via webhooks on the code repository or some other automated continuous integration (CI) process) automatic assembly of a new image from available artifacts, including the S2I base image, and the newly committed code.
  - Automatically deploys the newly built image for testing.
  - Can promote the tested image to production status and automatically deploy the new image through the CI process.



Red Hat OpenShift includes an integrated instance of Jenkins for CI. OpenShift also includes rich RESTful APIs that you can use to integrate your own build or CI tools or private image registry, such as JFrog's Artifactory.

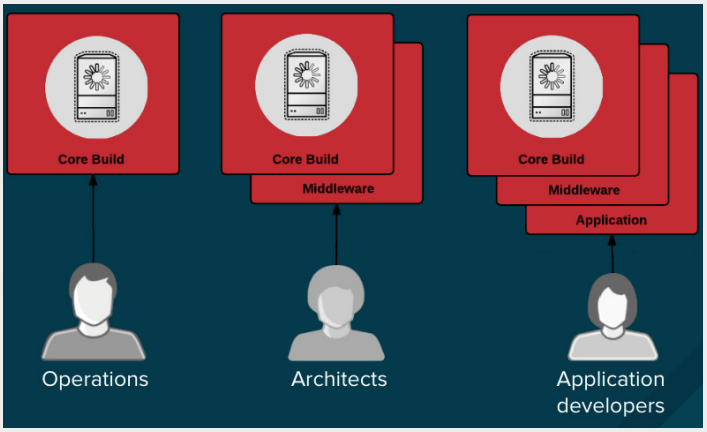
A best practice for application security is to integrate automated security testing into your build or CI process. For example, integrate:

- Static Application Security Testing (SAST) and Dynamic Applications Security Testing (DAST) tools like HP Fortify and IBM AppScan.
- Scanners for real-time checking against known vulnerabilities like Black Duck Hub and JFrog Xray. Tools like these 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 that your team can take appropriate action to address those issues sooner rather than later.

Whether you work in a highly regulated industry or simply want to optimize your team's efforts, we recommend that you design your container image management and build process to take advantage of container layers to implement separation of control, so that your:

- Operations team manages base images.
- Architects manage middleware, runtimes, databases, and other such solutions.
- Developers focus on application layers and just write code.



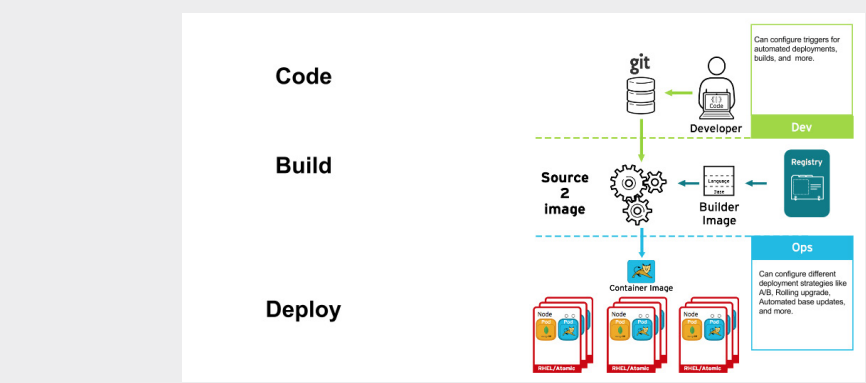
Finally, we recommend that you sign your custom-built containers so that you can be sure they are not tampered with between build and deployment.

5. Deployment: Controlling what can be deployed within a cluster

In case anything falls through during the build process, or for situations where a vulnerability is discovered after an image has been deployed, you want yet another layer of security: tools for automated, policy-based deployment.

Let's look at an application that's built using three container image layers: core, middleware, and the application layer. An issue is discovered in the core image and that image is rebuilt. Once the build is complete, the image is pushed to OpenShift's registry. OpenShift can detect that the image has changed. For builds that are dependent on this image, and that have triggers defined, OpenShift will automatically rebuild the application image, incorporating the fixed libraries.

Once the build is complete, the image is pushed to OpenShift's internal registry. 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.



Strong security includes automated policies that you can use to manage container deployment from a security point of view. OpenShift comes with built-in Security Context Constraints (SCCs) that you can use to define a set of conditions that a pod (a collection of containers) must run with in order to be accepted into the system. OpenShift's Security Context Constraints (SCCs), which have been contributed to Kubernetes as Pod Security Policy, allow an administrator to control the:

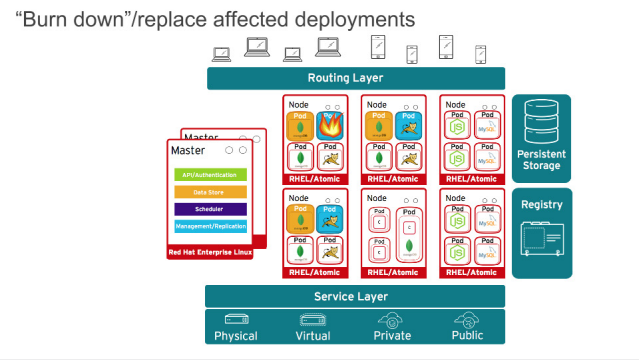
- Running of privileged containers.
- Capabilities a container can request to be added.
- Use of host directories as volumes.
- SELinux context of the container.
- Creation of seccomp profiles.
- Container user ID.

Users with the required permissions can adjust the default SCC policies to be more permissive if they choose. As with CI tooling, users can integrate their own CD tooling with their container platform if they prefer.

By integrating CI/CD with OpenShift, the entire process of rebuilding the application to incorporate the latest fixes and testing, and ensuring that it is deployed everywhere within the environment, can be fully automated.

6. Container orchestration: Securing the container platform

Of course, applications are rarely delivered in a single container. Even simple applications typically have a front end, a back end, 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 this diagram.



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. How will they discover each other?
- How you control access to—and management of—shared resources, like network and storage.
- How you monitor container health.
- How you automatically scale application capacity to meet demand.
- How to enable developer self-service while also meeting security requirements.

You can build your own container management environment. But why spend your time building tools when you can deploy a container platform with built-in management and security features, thus allowing your team to focus their energies on building the applications that provide business value?

Red Hat OpenShift Container Platform delivers container orchestration, automation of scheduling, and running application containers on clusters of physical or virtual machines through inclusion and extension of the open source Kubernetes project. Kubernetes, an open source project started by Google, uses "masters" to manage the complexity of container cluster orchestration. OpenShift also comes with Red Hat CloudForms which, among other things, can be used to monitor the health of containers in your private registry and prevent deployment of containers with newly detected vulnerabilities.

Given the wealth of capabilities for both developers and operators, strong role-based access control is a critical element of the container platform. For example, the orchestration masters are a central point of access and should receive the highest level of security scrutiny. APIs are key to automating container management at scale. APIs are used to validate and configure the data for pods, services, and replication controllers; perform project validation on incoming requests; and invoke triggers on other major system components.

API access control (authentication and authorization) is critical for securing your container platform. The OpenShift master includes a built-in OAuth server. Developers and administrators obtain OAuth access tokens to authenticate themselves to the API. As an administrator, you can configure OAuth to authenticate using the identity provider of your choice, including Lightweight Directory Access Protocol (LDAP) directories.

One of the key values of a container platform is the ability to enable developer self-service, making it easier and faster for your development teams to deliver applications built on approved layers. Multitenancy security is a must for the platform itself to make sure teams do not access each other's environments without authorization. You need a self-service portal that gives enough control to teams to foster collaboration while still providing security. OpenShift adds several components to Kubernetes to maintain a secure multitenant master, ensuring that:

- All access to the master is over transport layer security (TLS).
- Access to the API server is X.509 certificate- or token-based
- Project quota is used to limit how much damage a rogue token could do.
- Etcd is not exposed directly to the cluster.

OpenShift 3.5 delivers enhanced cluster management, including improved secrets and certificate management.

7. Networking isolation

Deploying modern microservices-based applications in containers often means deploying multiple containers distributed across multiple nodes. With network defense in mind, you need a way to isolate applications from one another within a cluster.

A typical public cloud container service, like Google Container Engine (GKE), Azure Container Services, or Amazon Web Services (AWS) Container Service, are single tenant services. They let you run your containers on the VM cluster that you initiate. For secure container multitenancy, you want 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.

With 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. Pods from different namespaces (projects) cannot send packets to or receive packets from pods and services of a different project by default, with exception options noted below. You can use these features to isolate developer, test, and production environments within a cluster.

However, this proliferation of IP addresses and ports makes networking more complicated. In addition, containers are designed to come and go. We recommend investing in tools that handle this complexity for you. A container platform that uses software defined networking (SDN) to provide a unified cluster network that enables communication between containers across the cluster is preferred.

Also preferred is a container platform that provides the ability to control egress traffic using either a router or firewall method so that you can use IP whitelisting to control, for example, database access.

In addition to network namespaces, the SDN provides additional security by offering isolation between master (orchestration) namespaces with the ovs-multitenant plug-in. When the ovs-multitenant plug-in is enabled, pod traffic within a namespace is, by default, isolated from pod traffic from another namespace (project). To provide exceptions for the ovs-multitenant plug-in, the "oadm pod-network" functionality was introduced in Red Hat OpenShift Container Platform 3.1 to allow two projects to access each other's services, or allow all projects to access all pods and services in the cluster. The limitation of this is that it operates at the level of the entire project, and the traffic allowed is always bidirectional. That is, if you can access a service in a project, you can access every service in that project, and you also necessarily granted access to all of your project's services. Because the permissions are bidirectional, this can only be configured by a cluster admin.

Introduced as a Tech Preview in Red Hat OpenShift Container Platform 3.5, a new Network Policy plug-in (ovs-networkpolicy) is designed to improve upon the way the ovs-multitenant plug-in can be used to configure allowable traffic between pods. Network Policy allows configuration of isolation policies at the level of individual pods. Because Network Policies are not bidirectional and apply only to ingress traffic of pods within a project administrator's control, cluster admin privileges are also not required.

If you wish to deploy network scanners, these are quite easy to containerize and can be run as "super privileged containers."

8. Storage

Containers are useful for both stateless and stateful applications. Protecting attached storage is a key element of securing stateful services. Red Hat OpenShift Container Platform provides plug-ins for multiple flavors of storage, including network file systems (NFS), AWS Elastic Block Stores (EBS), GCE Persistent Disks, GlusterFS, iSCSI, RADOS (Ceph), and Cinder.

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 (NFS, Ceph, Gluster, etc.), the trick is to have the shared storage 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 (EBS, GCE Persistent Disks, iSCSI, etc.), container platforms can use SELinux capabilities to secure the root of the mounted volume for nonprivileged pods, making the mounted volume owned by, and only visible to, the container it is associated with.

Data in transit should be encrypted via https for all container platform components communicating between each other.

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

9. API management / end point security and single sign-on (SSO)

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

Web 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, such as Red Hat SSO (RH-SSO), a fully supported, out-of-the-box SAML 2.0 or OpenID Connect-based authentication, web single sign-on, and federation service based on the upstream Keycloak project. RH-SSO 7.1 features client adapters for Red Hat JBoss Fuse and Red Hat JBoss Enterprise Application Platform (JBoss EAP). RH-SSO 7.1 includes a new Node.js client adapter, which enables authentication and web single sign-on for Node.js applications. RH-SSO can be integrated with LDAP-based directory services, including Microsoft Active Directory and Red Hat Enterprise Linux Identity Management. RH-SSO also integrates with social login providers such as Facebook, Google, and Twitter.

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. In addition, we recommend using an API management tool. 3Scale by Red Hat 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. These options include standard API keys, application ID and key pair, and OAuth 2.0.

3Scale's access control features go beyond basic security and authentication. Application and account plans let you restrict access to specific endpoints, methods, and services and apply access policy for groups of users. Application plans allow you to set rate limits for API usage and control traffic flow for groups of developers. Set per-period limits for incoming API calls to protect your infrastructure and keep traffic flowing smoothly. Automatically trigger overage alerts for applications that reach or exceed rate limits, and define behavior for over-limit applications.

10. Roles and access management in a cluster federation

In July of 2016, Kubernetes 1.3 introduced Kubernetes Federated Clusters for the first time. This is one of the exciting new features evolving in the Kubernetes upstream, currently in beta in Kubernetes 1.6. Federation is useful for deploying and accessing application services that span multiple clusters, running in the public cloud or enterprise datacenters. Multiple clusters can be useful to enable application high availability across multiple availability zones or to enable common management of deployments or migrations across multiple cloud providers, such as AWS, Google Cloud, and Azure.

When managing federated clusters, you will now need to be sure that your orchestration tools provide the security you need across the different deployment platform instances. As always, authentication and authorization are key—as well as the ability to securely pass data to your applications, wherever they run, and manage application multitenancy across clusters. Kubernetes is extending Cluster Federation to include support for Federated Secrets, Federated Namespaces and Ingress objects.

Federated secrets automatically creates and manages secrets across all clusters in a federation, ensuring that these are kept globally consistent and up to date, even if some clusters are offline when the original updates are applied.

Federated namespaces are similar to the traditional Kubernetes namespaces and provide the same functionality. Creating namespaces in the federation control plane ensures that they are synchronized across all the clusters in the federation.

Red Hat is working with the community and will add these capabilities to OpenShift as they mature.

CONCLUSION

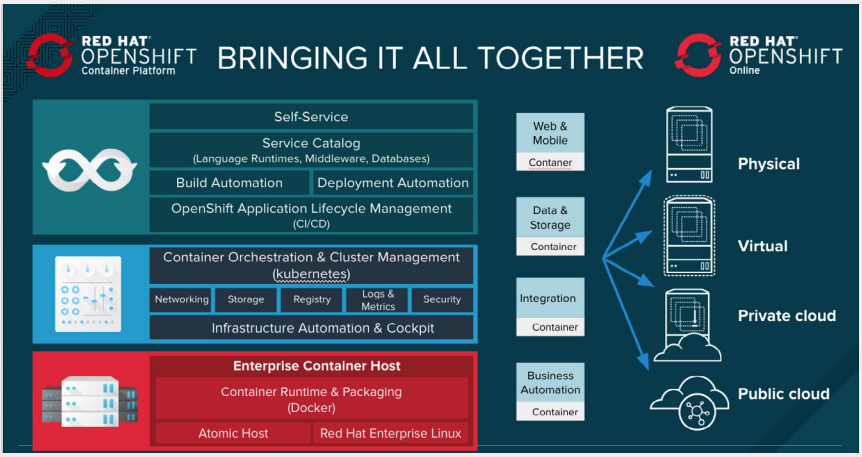
Of course, it is not just about security. Your container platform needs to provide an experience that works for your developers and your operations team. You need a secure, enterprise-grade container-based application platform that enables both developers and operators, without compromising the functions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

OpenShift 3 is built on a core of standard and portable Linux containers. OpenShift delivers built-in security features including:

- Strong, role-based access controls, with integrations to enterprise authentication systems.
- Powerful, web-scale container orchestration and management with Google Kubernetes.
- Integrated Red Hat Enterprise Linux 7 and Red Hat Enterprise Linux Atomic Host, optimized for running containers at scale with SELinux enabled for strong isolation.
- Integration with public and private registries.
- Integrated CI/CD tools for secure DevOps practices.
- A new model for container networking.
- Support for remote storage volumes.

OpenShift also provides the largest collection of supported programming languages, frameworks, and services. And support for federated clusters is on the roadmap.

OpenShift is available to run on OpenStack, VMware, AWS, GCP, Azure, and any platform that delivers Red Hat Enterprise Linux 7. Red Hat also provides OpenShift Dedicated and OpenShift Online as public cloud services.



As a leading provider of trusted, secure, and fully open source solutions to enterprise customers for the past 15+ users, Red Hat is now bringing this same level of trust and security to containers through solutions like Red Hat Enterprise Linux, Red Hat OpenShift Container Platform, and our entire container-enabled Red Hat product portfolio.

ABOUT RED HAT

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.co

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

redhat.com
#f7530_0517