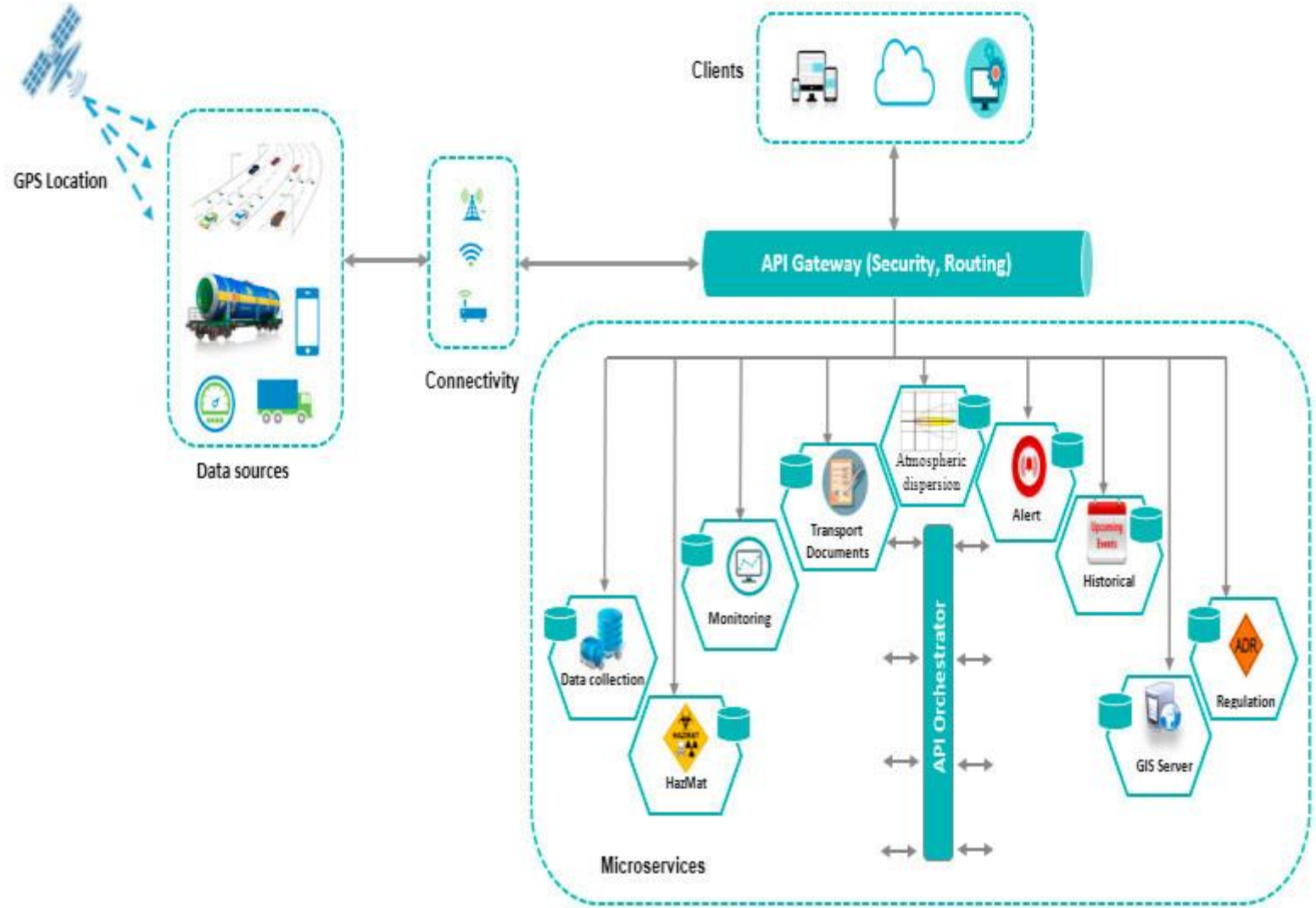
IBM MICROSERVICE I
SESSION 2

TEXAS SKILLS DEVELOPMENT FUND
TRAINING PARTNERSHIP

AUSTIN COMMUNITY COLLEGE DISTRICT

Security at Scale

- 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 to control access and management of shared resources?
  - For example - network and storage
- How to monitor container health?
- How to automatically scale application capacity to meet demand?
- How to enable developer self-service while also meeting security requirements?



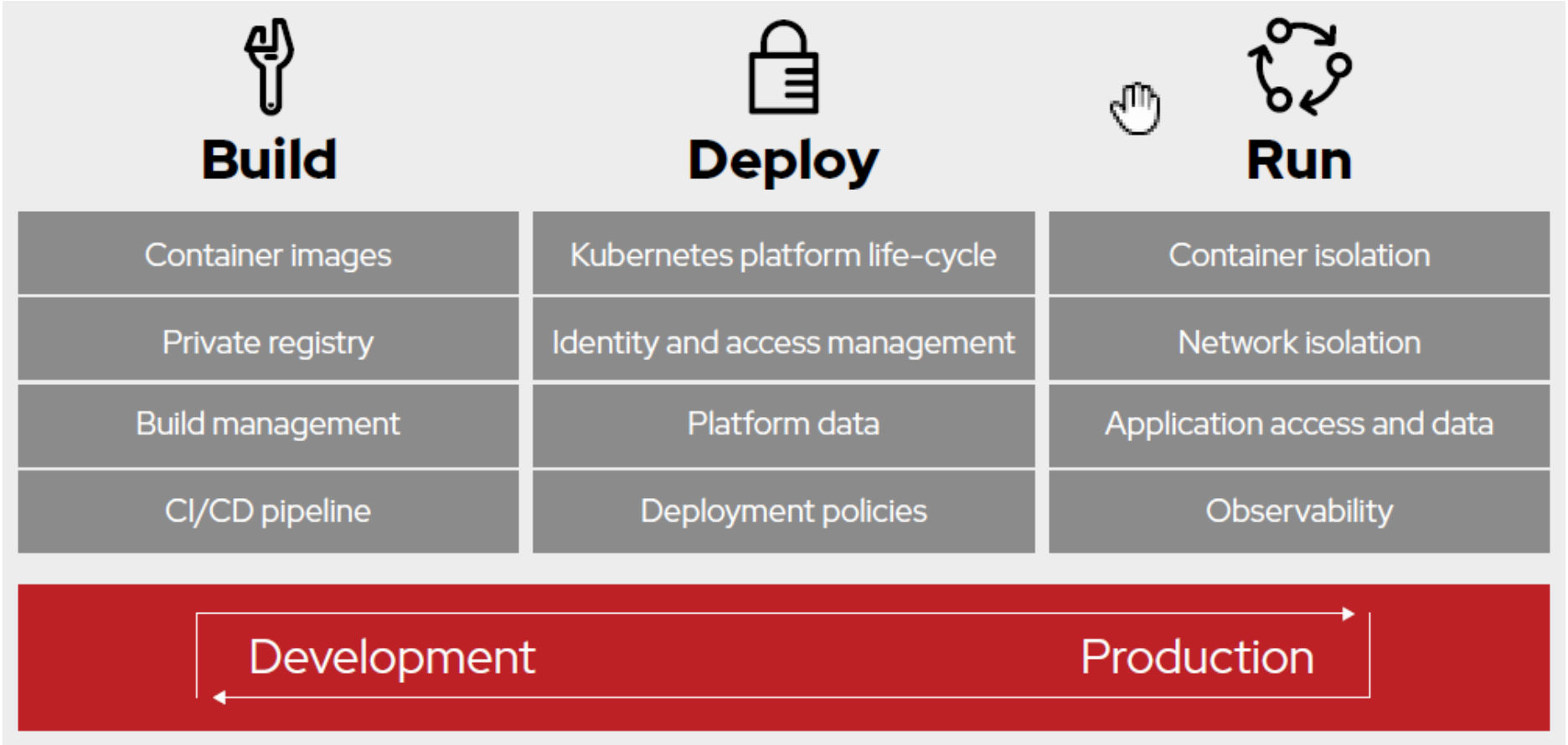
Real-Time HazMat Environmental Information System: A micro-service based architecture

Container & Kubernetes Security – Layered Approach

Layers & lifecycle, **Build** security, **Deployment** security & compliance

Runtime protection, Secure ecosystem

Container Security



A layered approach to container and Kubernetes security

Secure Build

Secure Container Build

Use trusted container content

- Red Hat Universal Base Image provides greater reliability, security, and performance for OCI-compliant images
- Red Hat Ecosystem Catalog
  - Certified images and operators for various language runtimes, middleware, databases for bare metal to VMs to cloud
- Container Health Index
- Container scanning tools

CONTAINER SECURITY PROVIDERS	
Aqua Security	Portshift
Capsule8	Stablewave
Cloudentity	StackRox
Layered Insight	Tenable
NeuVector	Twistlock

Secure Container Build

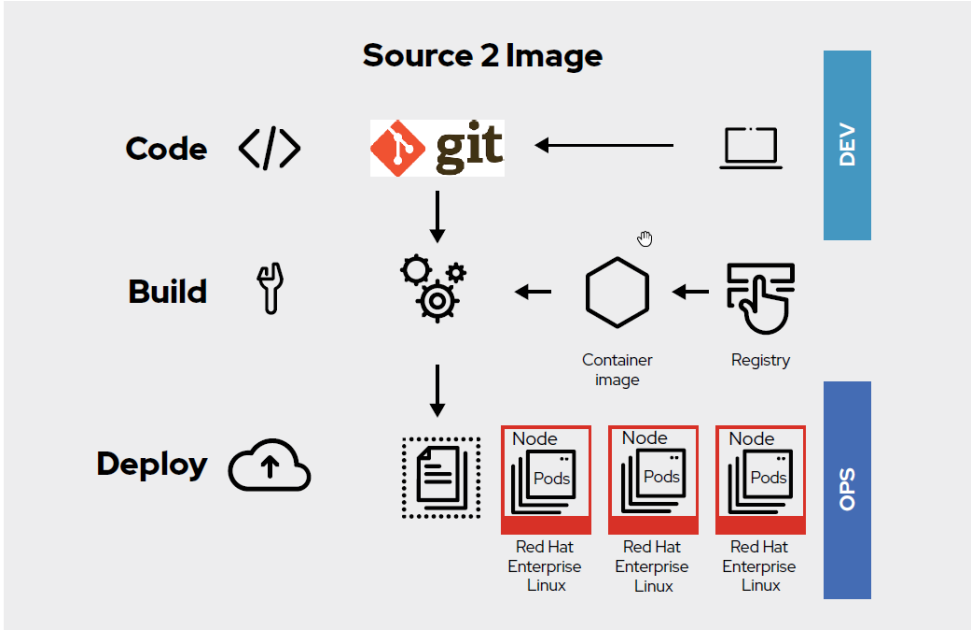
Use an enterprise container registry

- Red Hat OpenShift registry provides role-based access control (RBAC)
- Supports integration with other private registries, e.g. JFrog's Artifactory, Sonatype Nexus
- Red Hat Quay is available as a standalone enterprise registry
  - Enterprise features - geographic replication, image triggers
- Clair project - open source engine that powers the Red Hat Quay security scanner
  - Red Hat OpenShift Container Security Operator integrates with Red Hat Quay to provide a cluster-wide view of known vulnerabilities for OpenShift deployed containers

Secure Container Build

Control & automate building containers

- 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) open source framework:
- 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.



Secure Container Build

Control & automate building containers

- Red Hat OpenShift image streams can be used to watch changes to external images deployed into a 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

Secure Container Build

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 to integrate 3rd party build or CI/CD tools including a private image registry.
- A best practice for application security is to integrate automated security testing into the DevOps pipeline, including registry, integrated development environment (IDE), and your CI/CD tools

Secure Container Build

Integrate security into the application pipeline

- Registry
  - Container images can and should be scanned in the private container registry
  - 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
- 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 any container, alert known vulnerabilities, and update when new vulnerabilities are discovered in previously scanned packages.
- Additionally, the CI process should include policies that flag builds with issues discovered by security scans so the team can take appropriate action to address those issues as soon as possible.
- Custom built containers should be digitally signed to ensure integrity between build and deployment.

Deployment Security

Managing the configuration, security, and compliance

Secure Container Deployment

Platform configuration and life cycle management

- CNCF Kubernetes Security Audit (2019)
  - 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
- Kubernetes Operator
  - Method of packaging, deploying, and managing a Kubernetes-native application
  - 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 configure components - API server, SDN
  - Cluster version operator manages multiple operators across the platform
- Operators enable automatic cluster management, including updates, from the kernel to services higher in the stack

Secure Container Deployment

Platform configuration and life cycle management

- Red Hat OpenShift cluster Operator Lifecycle Manager (OLM) framework – DevOps Self Service
  - Find and use operators to deploy the services needed to enable their applications
  - Install, upgrade, and assign role-based access control to available operators
- Compliance Operator
  - Automate the platform's compliance with technical controls required by compliance frameworks
  - Admins can describe the desired compliance state of a cluster
  - Provides an overview of gaps and remediation steps
  - Assesses compliance of all platform layers, including the nodes running the cluster.
- File Integrity Operator
  - Available to run file integrity checks on the cluster nodes regularly

Secure Container Deployment

Identity and access management

- The API server (for Kubernetes API) is a central point of access and should receive the highest level of security scrutiny
- Red Hat OpenShift control plane includes built-in authentication through the Cluster Authentication operator
  - Exchange OAuth access tokens with a variety of identity providers
  - Cluster administrators can use the cluster roles and bindings to control access levels to the OpenShift cluster and to projects within the cluster

Secure Container Deployment

Securing platform data and attached storage

OpenShift Data Protection in Transit Features:

- 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

Secure Container Deployment

Securing platform data and attached storage

OpenShift Data Protection at Rest Features:

- 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
- Optionally enable FIPS-140-2 mode in the RHEL CoreOS
- Red Hat OpenShift platform components call Red Hat Enterprise Linux cryptographic modules
- Red Hat OpenShift supports both ephemeral and persistent storage
    - Protecting attached storage is a key element of securing stateful services

Secure Container Deployment

Securing platform data and attached storage

OpenShift Data Protection at Rest Features:

- 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.
  - 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
    - ReadWriteOnce, ReadOnlyMany, and ReadWriteMany.

Secure Container Deployment

Securing platform data and attached storage

OpenShift Data Protection at Rest Features:

- For **shared storage** (e.g., NFS, Ceph, Gluster), the shared storage persistent volume (PV) registers 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

Secure Container Deployment

Deployment policies

Policy-based container deployment

- Red Hat OpenShift clusters can be configured to allow or disallow images to be pulled from specific image registries
  - Best Practice - Production clusters should only allow images to be deployed from the designated 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 (SCC)** drop privileges (default best practice)
  - Ensure that 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
- Red Had Advanced Cluster Management for Kubernetes provides **advanced application life-cycle management** using open standards to deploy applications
  - Placement policies integrated into existing CI/CD pipelines and governance controls

Secure Container Deployment

Deployment policies

Policy-based multi-cluster deployment

- Provide application high availability across multiple availability zones or functionality for common management of deployments or migrations across multiple cloud providers
- Red Had Advanced Cluster Management for Kubernetes
  - Multi-cluster life-cycle management 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
  - Also specify a compliance policy to apply across one or more managed clusters

Runtime Protection

Protect Running Applications

Runtime Protection

Container isolation

- 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
- NIST special publication 800-190 recommends using a container-optimized OS for additional security
- Red Hat Enterprise Linux CoreOS reduces the attack surface by minimizing the host environment and tuning it for containers
  - Only contains the packages necessary to run Red Hat OpenShift and its userspace is read-only
  - Tested, versioned, and shipped in conjunction with Red Hat OpenShift and it is managed by the cluster
  - Installation and updates are automated and always compatible with the cluster
  - Supports the infrastructure of choice, inheriting most of the RHEL ecosystem
  - Every Linux container running on a Red Hat OpenShift platform is protected by powerful RHEL 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

Runtime Protection

Container isolation

- 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

Runtime Protection

Container isolation

- 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 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 infrastructure activities, e.g., sending raw internet protocol (IP) packets or bind to ports below 1024
  - When running containers, 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

Runtime Protection

Application and network isolation

- 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 used to control access to individual projects
- **Security context constraints** drop privileges by default (important 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

Runtime Protection

Application and network isolation

- Container platform must segment the traffic to isolate different users, teams, applications, and environments within a cluster
- Also need to manage external access to the cluster and access from cluster services to external components
- **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

Runtime Protection

Application and network isolation

- **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
    - IP whitelisting to provide access to an external database

Runtime Protection

Securing Application Access

- **Controlling user access**
- 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
  - 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
  - Also integrates with social login providers such as Facebook, Google, and Twitter

Runtime Protection

Securing Application Access

- **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.
- Red Hat 3scale API Management
  - Standard options for API authentication and security that can be used alone or in combination to issue credentials and control access
  - Application and account plans - restrict access to specific endpoints, methods, and services
  - Apply access policies for groups of users, set rate limits for API usage and control traffic flow
  - Set per-period limits for incoming API calls to protect infrastructure smooth traffic flow
  - Automatically trigger overage alerts for applications that reach or exceed rate limits
  - Define behavior for over-limit applications.

Runtime Protection

Securing Application Access

- **Securing Application Traffic**
- For microservice-based applications, security traffic between services on the cluster is important. A service mesh can be used to deliver this management layer.
  - “service mesh” describes the network of microservices that make up applications in a distributed microservice architecture and the interactions between those microservices.
- 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.
  - Uses a multitenant operator to manage the control plane life cycle; used on a per-project basis.
  - Does not require cluster-scoped RBAC resources.
  - 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.

Runtime Protection

Observability

Prometheus Monitoring Ecosystem

- Built-in monitoring and auditing as well as an optional logging stack with alert dashboard
- 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
- Red Hat OpenShift auditing was designed using a cloud-native approach to provide both centralization and resiliency.
- Host auditing and event auditing are enabled by default on all nodes
- Supports flexibility for configuring management and access to auditing data.
- Control the amount of information in audit logs with audit log policy profile

Runtime Protection

Observability

Prometheus Monitoring Ecosystem

- Monitoring, audit, and log data is RBAC-protected
- Project data is available to project administrators and cluster data is available to cluster administrators.
- Best Practices - configure 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 cluster nodes and stores them in a default log store
- Multiple options are available for forwarding logs to chosen SIEM platform

Secure Ecosystem

Secure Ecosystem

Privileged access management.

External certificate authorities.

External vaults and key management solutions.

Container content scanners and vulnerability management tools.

Container runtime analysis tools

SIEM