Red Hat Quay: Technical Overview and Roadmap

Laine Vyvyan
Channel Solutions Architect

Joshua Smith
Cloud Solutions Architect

Dirk Hermann
Senior Principal Product Manager, Quay

If you’re doing containers and Kubernetes, you need a container image registry

Red Hat

Just like money, containers need safe and reliable storage and transmission

Core Services of a Container Registry


Pushes (Deposits),
Pulls (Withdrawals),
and *Security*

What is Quay?

SCALABILITY

Massive Scale Testing Quay.io
Real Time Garbage Collection
Automated Squashing

BUILD AUTOMATION

Seamless Git Integration
Build Workers
Webhooks

INTEGRATION

Extensible API
Webhooks, OAuth
Robot Accounts

ENTERPRISE REGISTRY

High Availability
Full Standards / Spec Support
Long-Term Protocol Support
Application Registry
Enterprise Grade Support
Regular Updates

SECURITY

Vulnerability Scanning
Logging & Auditing
Notifications & Alerting

CONTENT DISTRIBUTION

Geo-Replication
Repository Mirroring
Air-Gapped Environments

ACCESS CONTROL

Authentication Providers
Fine-Grained RBAC
Organizations & Teams

What is Quay?

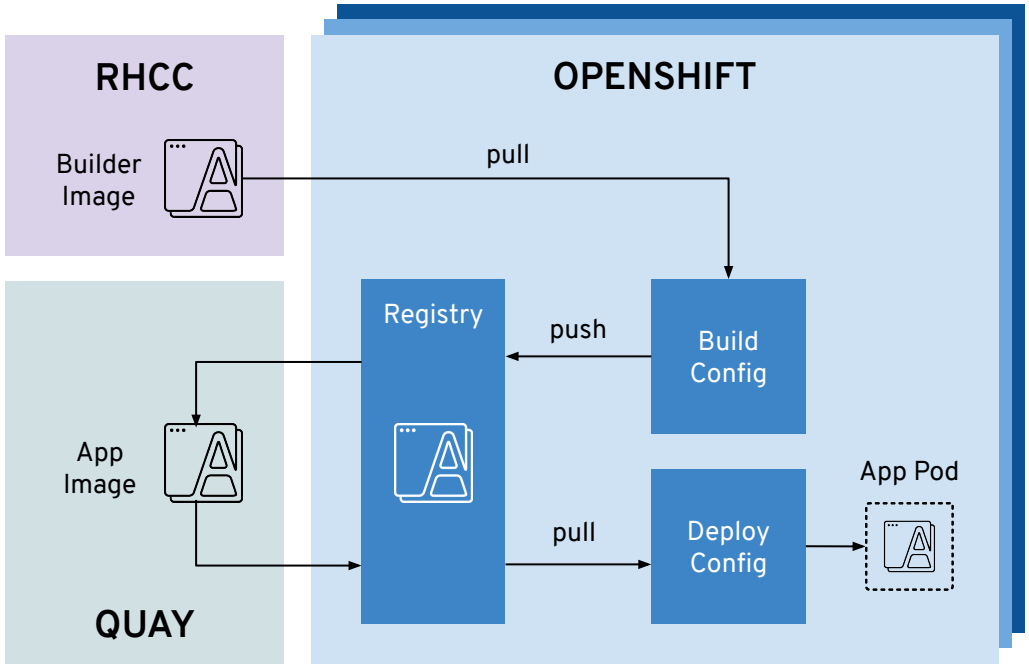
- Enterprise container registry
- Available on-premise, on public cloud, and as a hosted service (SaaS)
- Key strengths:
  - Security
  - Robustness & speed
  - Automation
- Quay works with any container environment or orchestration platform

First hosted registry in the market with private repos

One of the top 5 biggest hosted registries overall

Quay as an Upstream Registry with OpenShift

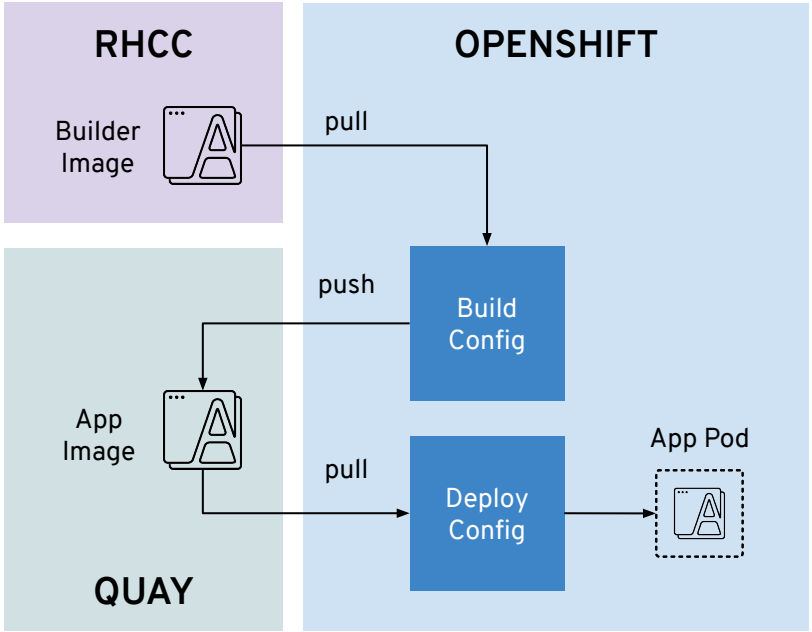
- Images **pulled** from Quay into the integrated OpenShift registry
- Images are **pushed** to the integrated OpenShift registry, and synced externally with Quay



Red Hat

Quay as the OpenShift Registry

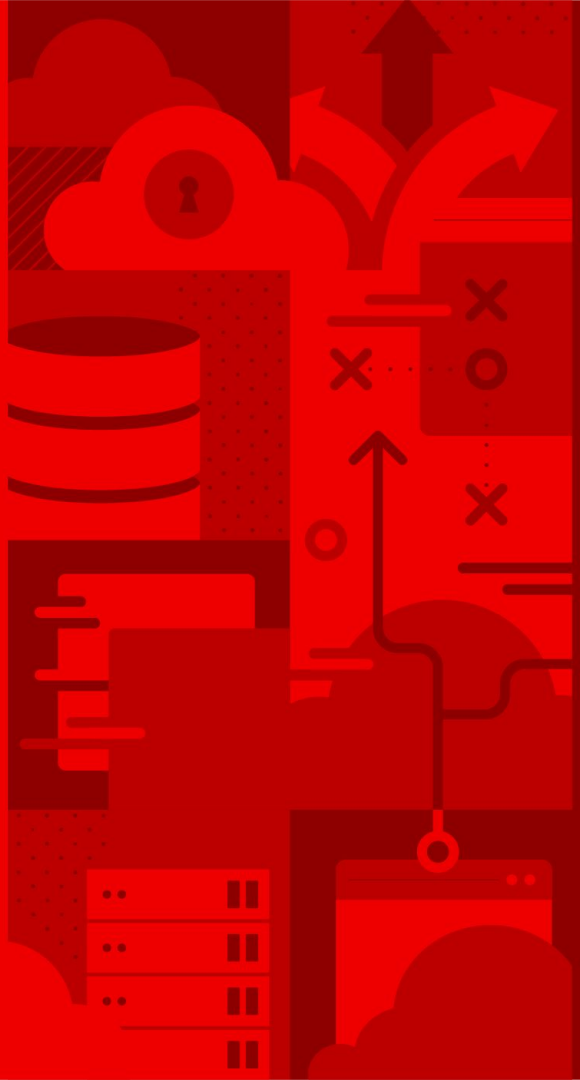
- Images are pushed directly by builds to Quay
- Images are pulled directly from Quay



Red Hat

Common Quay Use Cases

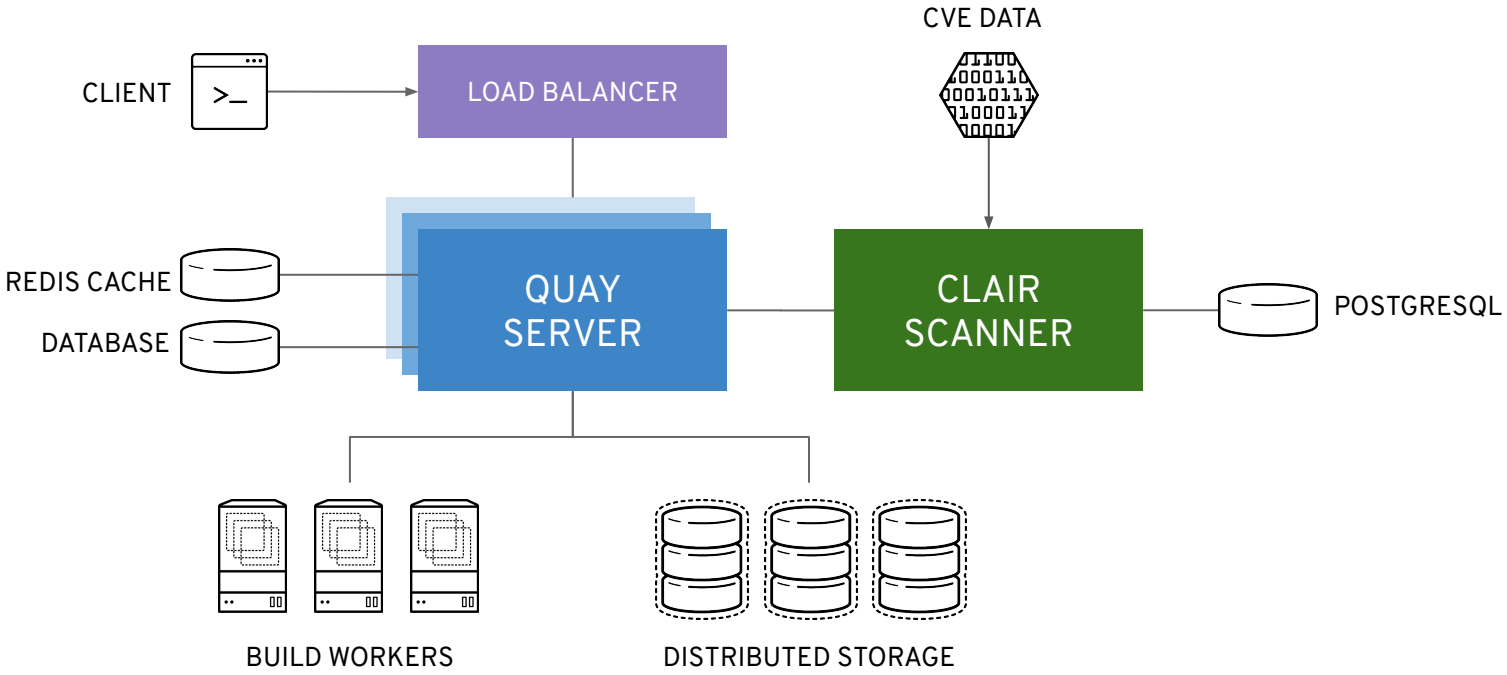
- Large-scale and distributed environments (thousands of users and images)
- Images shared across multiple OpenShift clusters
- OpenShift clusters in multiple geographical regions
- Governance/security of container images
- High image maintenance and automation requirements
- Large number of build and high requirements on image delivery throughput



Red Hat Quay Architecture

Red Hat

Red Hat Quay Architecture



Note: The Red Hat Quay subscription doesn't include several components shown here such as the storage or database backends, load balancer and redis cache.

Red Hat

Prerequisite #1: The Database

- **PostgreSQL** is the preferred database backend since it can be used for both Quay and Clair
- If Clair isn't used: Quay works fine with MySQL too (5.7+)
- **If Quay runs on public cloud infrastructure** we recommend to use the PostgreSQL services provided by your cloud provider
- If the database runs on **OpenShift** we recommend to use an operator, such as Crunchy Data PostgreSQL Operator

Full list of tested and supported configurations can be found inside the Red Hat Quay Tested Integrations Matrix: https://access.redhat.com/articles/4067991

Red Hat

Prerequisite #2: Storage Backend

Public Cloud Storage

- AWS S3
- Google Cloud Storage
- Azure Blob Storage

Microsoft Azure Blob Storage

On-Prem Storage

- Ceph Rados RGW
- OpenStack Swift
- RHOCS3 (via NooBaa) (TP)
- RHOCS4 (via NooBaa)

Red Hat OpenShift Container Storage

noobaa

Note: Local Storage for PoC's only, NFS not recommended

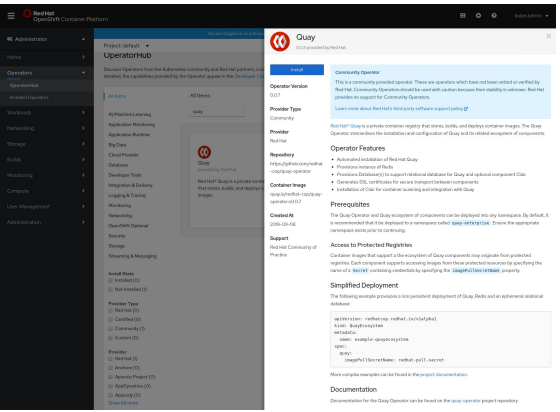
Full list of tested and supported configurations can be found inside the Red Hat Quay Tested Integrations Matrix: https://access.redhat.com/articles/4067991

Red Hat

Prerequisite #3: Redis Cache

- Provided via Red Hat Software Collections but any other redis works, too
- Mostly used by builds, workers and tutorial
- Data stored is ephemeral in nature, Redis does **not** need to be HA.
- If Redis goes down you will lose access to:
  - Live build logs
  - Tutorial

Quay Setup Operator



All Places > Communities at Red Hat > Applications

Container Community of Practice

Quay Setup Operator

Automates the initial deployment of Quay and Clair

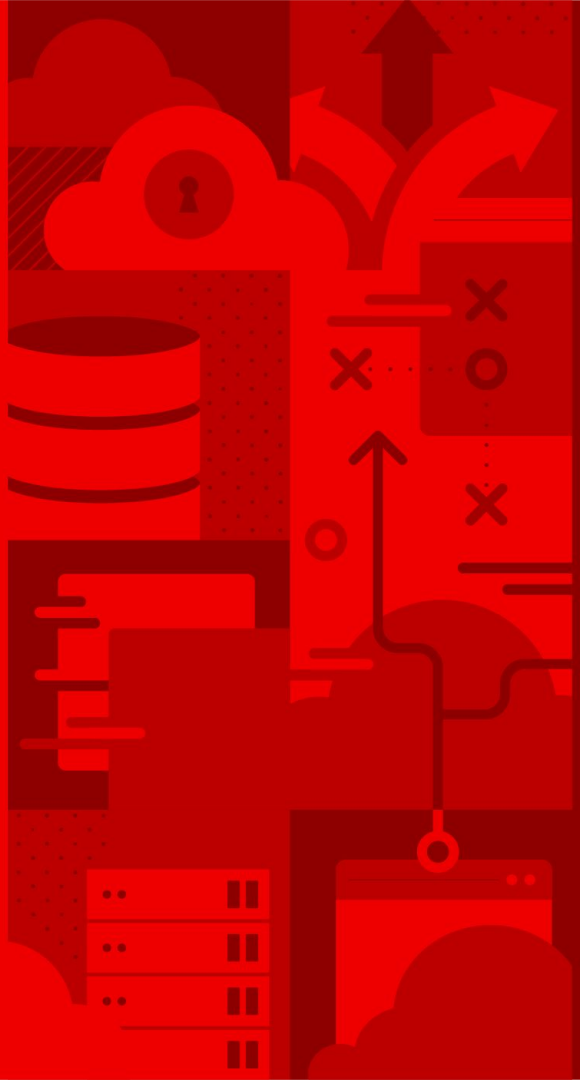
Simplifies installation of Quay

Configures all relevant OpenShift objects (routes, secrets, etc.)

Aids in certificate management

Check out the Quay Setup Operator Demo: https://youtu.be/TCDmyIt1Fns

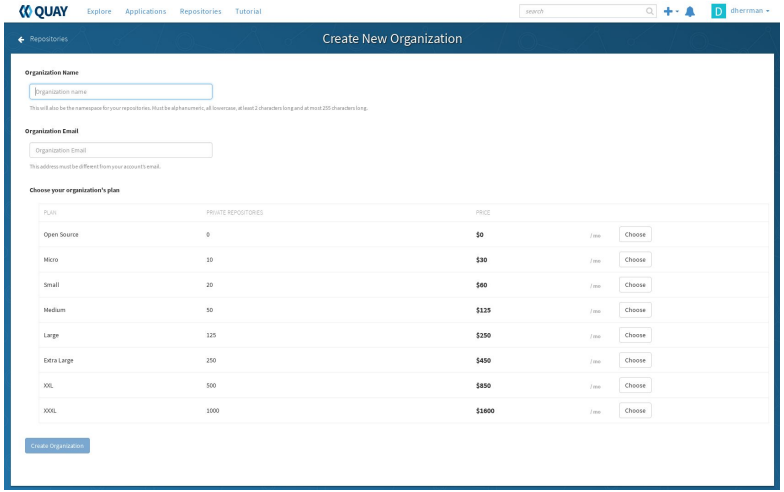
Red Hat



Authorization and Authentication

Organizations, Teams, Users, Robot Accounts

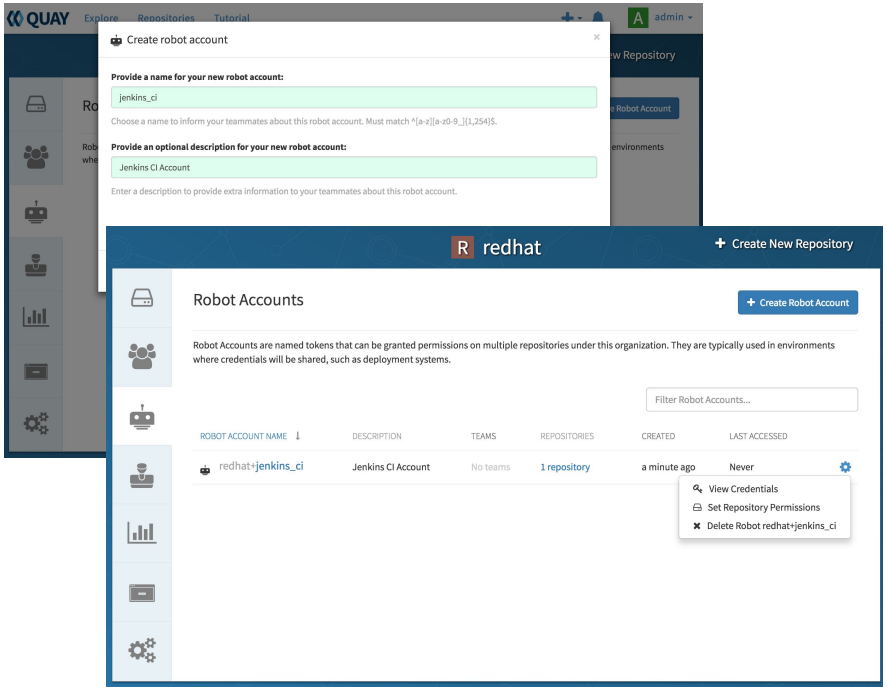
- Organizations
  - sharing repositories under a common namespace that belongs to many users
  - divided into **teams** which provide access to repositories under that namespace
- Teams
  - Provide a way for an organization to delegate permissions (both global and on specific repositories) to groups of users
  - Permissions: Member, Creator, Admin



Red Hat

Organizations, Teams, Users, Robot Accounts

- Users
  - Key element of setting repository permissions / RBAC
- Robot accounts
  - Allow for automatic software deployments
  - Can be shared by multiple repositories owned by a user or organization
  - Managed inside the organization view -> *Robot Accounts* tab



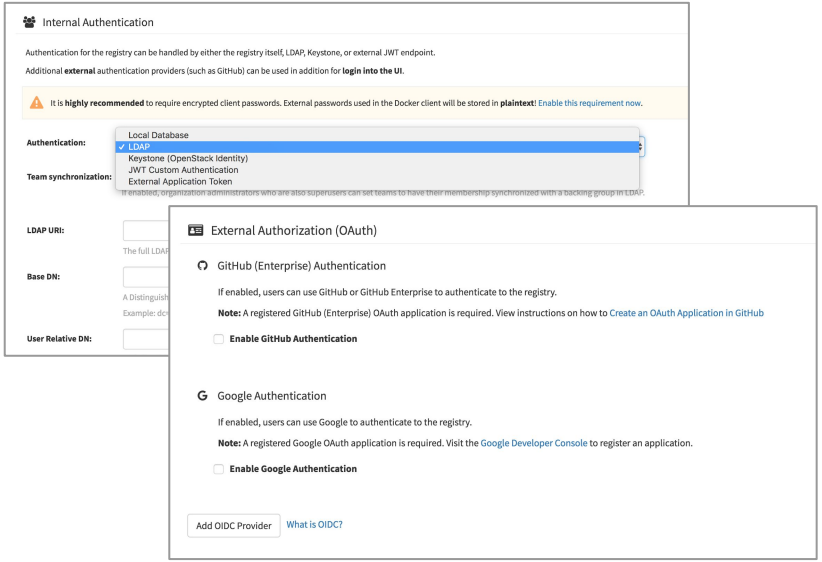
Red Hat

Enterprise Authorization and Authentication

Red Hat Quay allows you to integrate your existing identity infrastructure and use a fine-grained permissions system to map to your organizational structure and grant access to whole teams to manage specific repositories.

Supported auth providers:

- Built-in Database Auth
- LDAP auth and sync
- External OIDC provider
- OpenStack Keystone



Note: Auth integration for OCP coming soon (next slide…).

Red Hat

Quay and OpenShift Mapping (coming soon)

- Each OpenShift **namespace** results in an **organization** within Quay
- Each **ImageStream** within an namespace results in a new **repository** within Quay
- The three primary service accounts (Builder, Deployer and Default) result in **Robot accounts** in each organization with the following permissions
  - Builder - Write
  - Default - Read
  - Deployer - Read
- **Secrets** from each Robot account within an org are created in each OpenShift namespace
- **Service accounts** are configured to leverage these secrets as mountable secrets in order to push images as well as pull secrets in order to push images

Early prototype already exists developed in collaboration with our Community of Practice

https://github.com/redhat-cop/quay-openshift-registry-operator

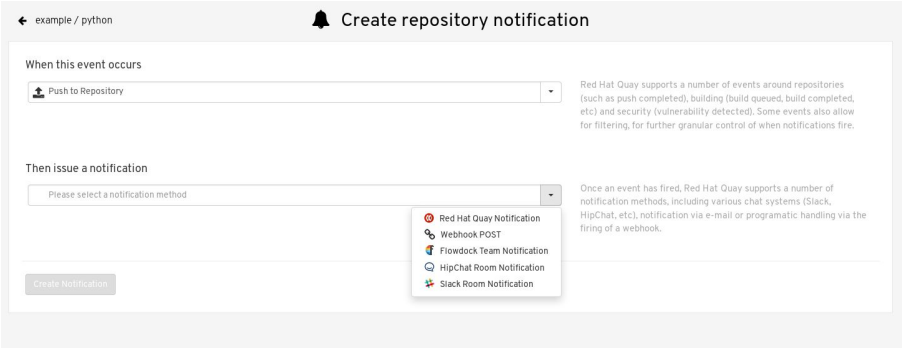
Repository Notifications

Quay triggers different notifications for various repository events.

E-mail, Quay UI notification, webhook POST (can be used to trigger other integrations - for example an OpenShift Pipeline or a Jenkins job), Flowdock, Hipchat, Slack notifications based on various events inside of Quay Enterprise.

How it Works:

- Events
- [Event threshold] (vulnerability: severity)
- Notification method



Red Hat

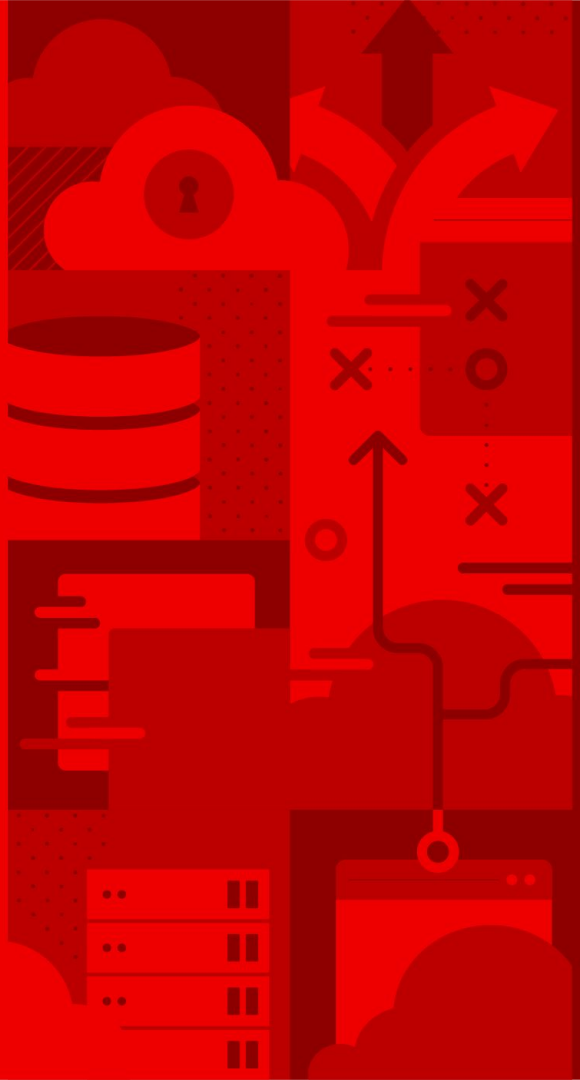


Image Management

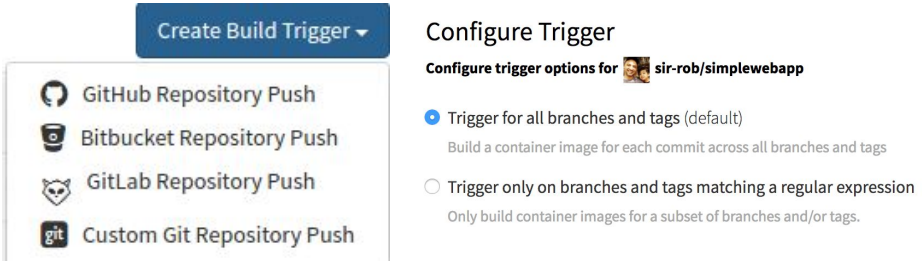
Red Hat

Quay Build Triggers

Automatically build images on repository push actions with integration to GitHub, Bitbucket, and more.

How it Works:

- Must be enabled in Admin panel first
- Mounts in docker.socket
- Equivalent to running `docker build`cmd
- Robot accounts to lock down automated access and audit each deployment



Note: Docker version must be the same across all build nodes!

Note: If self-signed certs are used, the cert must be mounted in via the configuration directory!

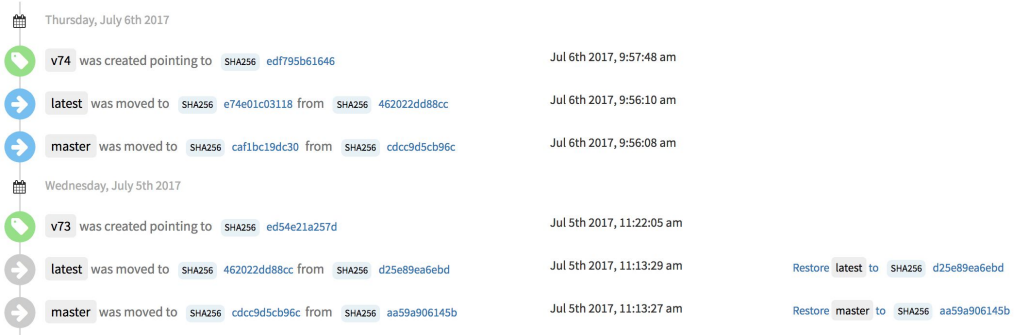
Red Hat

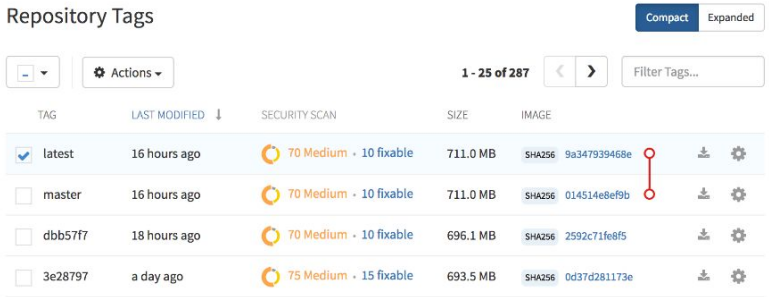
Time Machine

Time Machine provides image rollback: view history of tags for up to two weeks and have the ability to revert tags to a previous state.

How it Works:

- Select Repository → Click on Tags tab
- Add a new tag / move a tag / delete tag
- View tag history
- Click on Restore to roll back





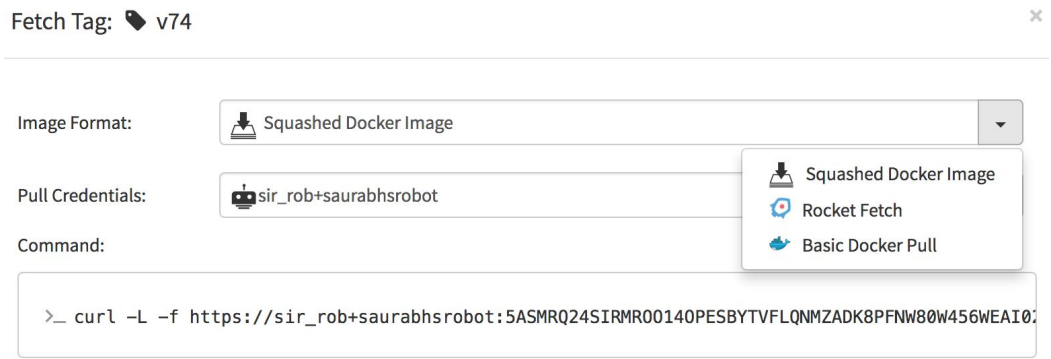
Red Hat

Automatically Squash Images

Squash multiple docker layers into one in order to create an image with fewer and smaller layers - leading to reduced image size and faster fetch times.

How it Works:

- HTTP call triggers squashing and fetches squashed image
- Resulting squashed image is stored in Quay but you can't *docker pull* it



Note: Images provided by Red Hat are squashed as part of our internal image build factory.

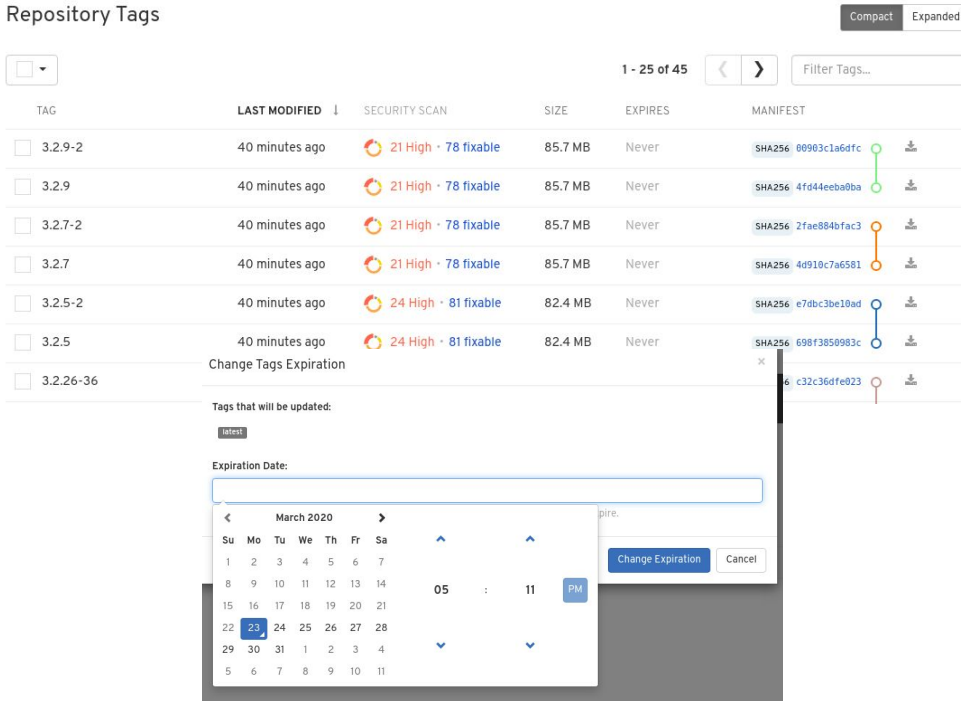
Red Hat

Tag Expiration (Retention Policy)

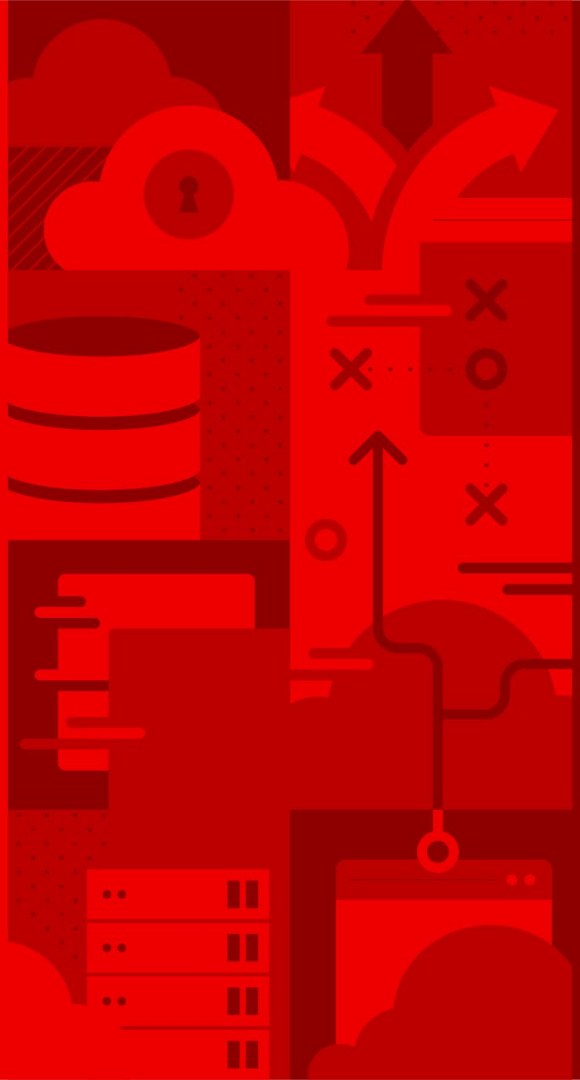
Users can add a `quay.expires-after=20h` label via the Dockerfile LABEL command. The values can be things like 1h, 2d, 3w for hour, day and weeks, respectively

Additionally, there's a column in the UI that can be set entitled “Expires.”

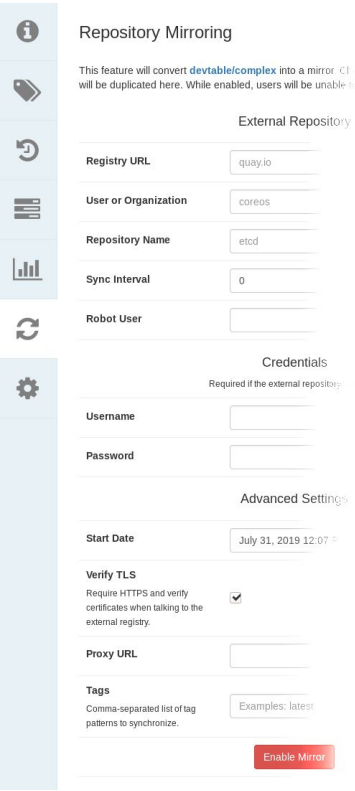
The tag/image version will be deleted from the repo when the expiration time is reached.

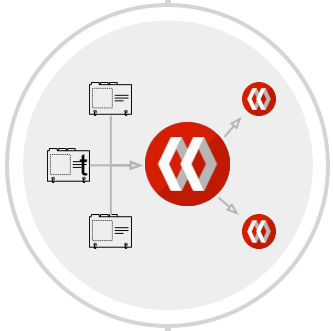


Red Hat



Repository Mirroring





Powered by

skopeo

https://github.com/containers/skopeo

Repository Mirroring

Allows to continually synchronize repositories from external source registries into Quay (content ingress point)

Allows to mirror a subset of the entire registry content to distributed registry deployments

Filters could be applied to sync only a subset of a repository

New: API calls available to automate mirroring configuration

Introduced as Dev Preview in already in Quay 3.1 - now **GA**

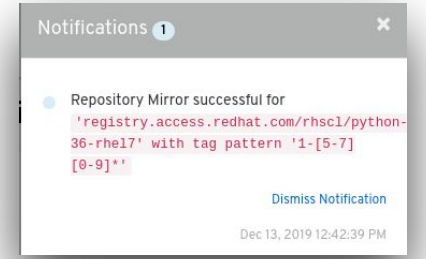
Red Hat

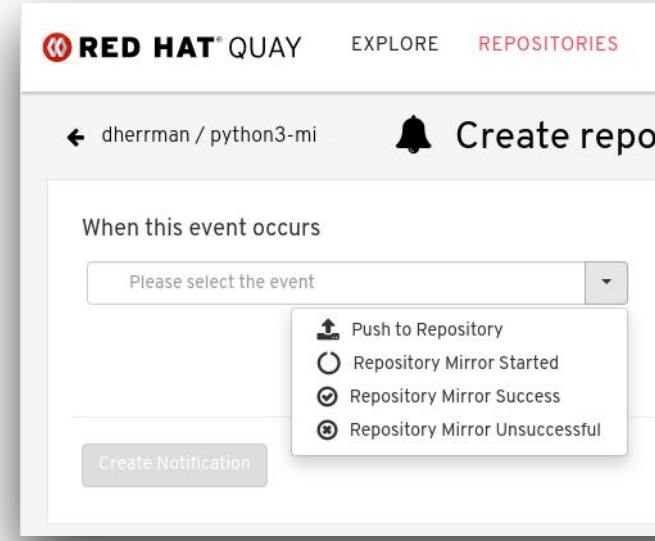
Notifications

In addition to pre-existing notification events we added 3 repo mirroring related events which can be configured inside the Settings tab for each repository.

New Notification Events:

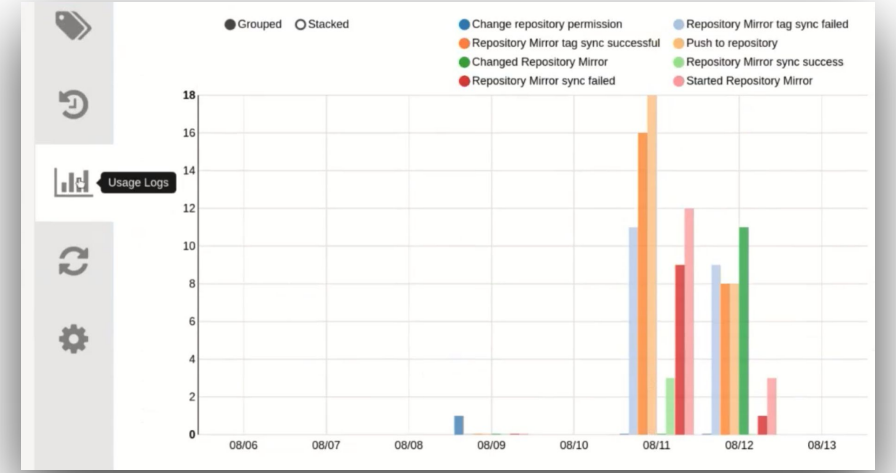
- Repository Mirror Started
- Repository Mirror Success
- Repository Mirror Unsuccessful

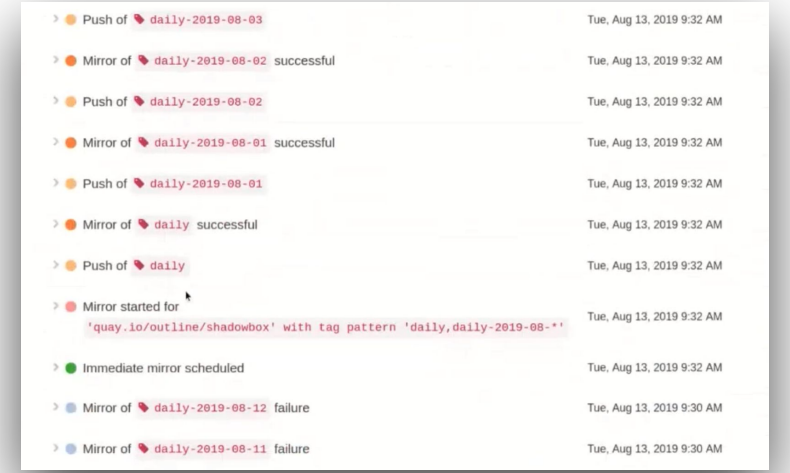




Red Hat

Logging and Auditing

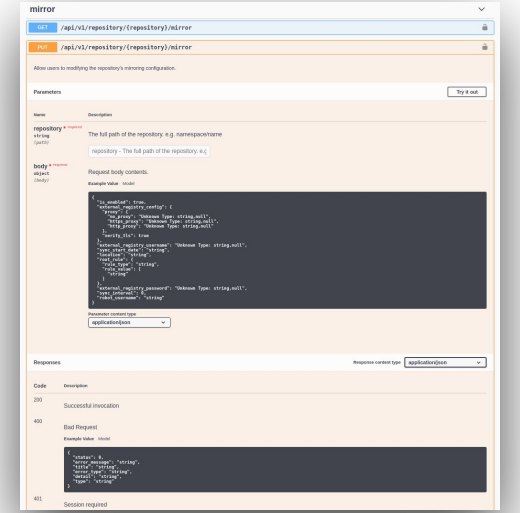




- Detailed auditing and logging including the skopeo log information

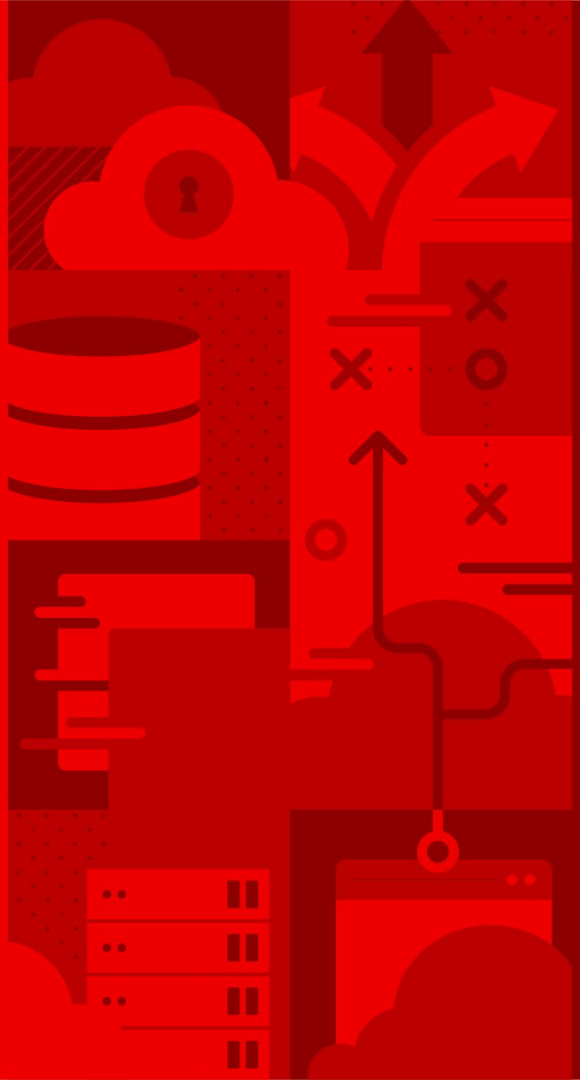
Red Hat

API Calls



- All repository mirroring configuration can be done via Quay API
- Documentation via Swagger UI (see Quay API section for further details)

Red Hat



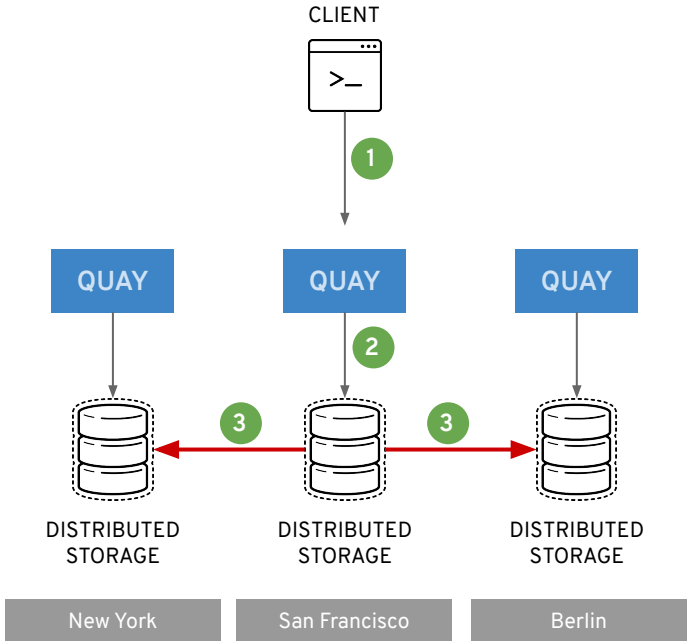
Geo-Replication

Quay Geo-Replication

Geo-replication allows for a single globally-distributed Quay Enterprise to serve container images from localized storage.

How it Works:

- Image data will be asynchronously replicated in the background
- By default *all* images are replicated to *all* storage engines configured

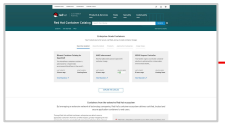


Red Hat

Multi-Region Quay Setup

- Geo-Replication means a **single, globally-distributed** Red Hat Quay setup spread across multiple datacenters / regions

Red Hat Container Catalog



Suppliers

Community

Custom

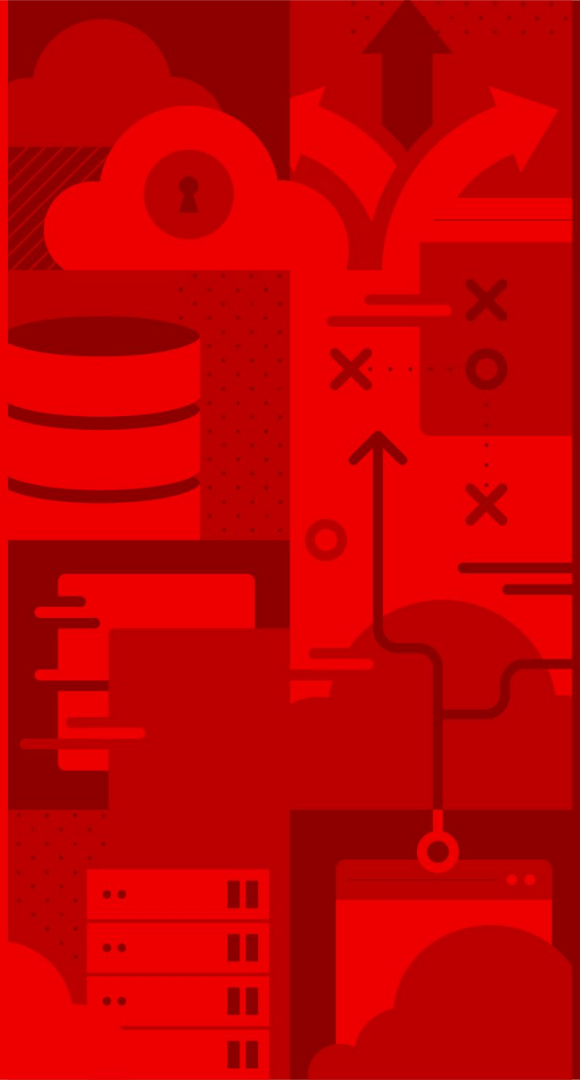
Primary Registry ("Single Source of Truth")

HA Setup
Geo-Replication
Webhooks

Primary Registry (Second Region)

- Clients pull from their preferred (localized) storage to access blobs
- Shared database -> all other operations are handled globally, frontend depends
- blob not replicated yet: fallback to remote

Red Hat



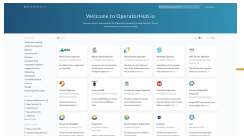
Repository Mirroring vs/*and* Geo-Replication

Quay Repository Mirroring vs. Geo-Replication

Feature / Capability	Geo-Replication	Repository Mirroring
Feature is designed for	a shared, global registry	Distinct, different registries
If replication / mirroring hasn't been completed yet then...	The remote copy is used (slower)	No image is served
Access to all storage backends in both regions required	Yes (all Quay nodes)	No (distinct storage)
Users can push images from both sites to the same repository	yes	no
whole registry content and configuration is identical across all regions (shared database)	yes	no
users can select individual namespaces / repositories to be mirrored	no	yes
users can apply filters (tag, tag range, etc.) to synchronization rules	no	yes
allows individual / different RBAC configurations in each region	no	yes

Content (Re)Distribution across the Globe

operatorhub.io *



Red Hat Container Catalog

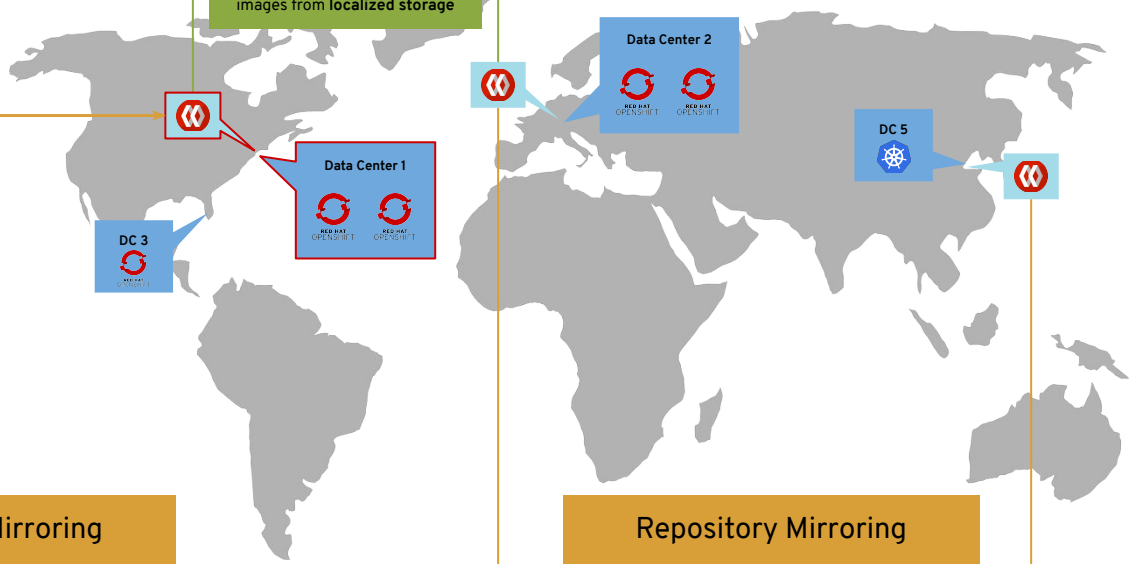
Suppliers

Community

Custom

Geo-Replication

single globally-distributed Red Hat Quay to serve container images from **localized storage**



Repository Mirroring

to mirror **external content into Quay**
explicit **whitelisting** / content **ingress**
supports **disconnected** clusters **

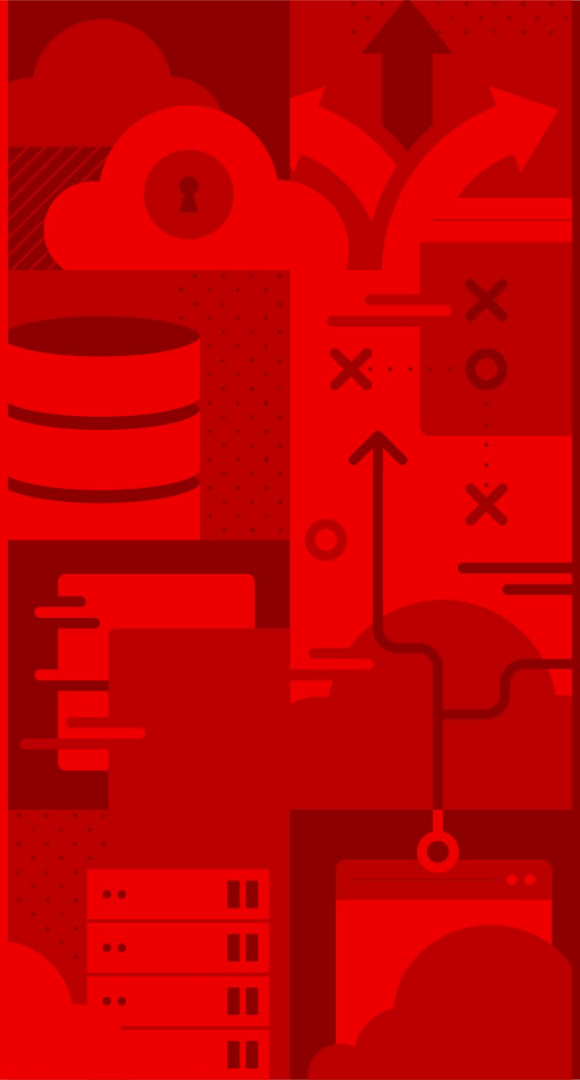
Repository Mirroring

to mirror a **specific subset**
of content to specific
data centers, clusters, regions

* Mirroring of operator repositories from operatorhub.io and expose them as a private catalogsource to the embedded operatorhub in OpenShift planned for future releases of Red Hat Quay

** Quay connected but clusters only have access to Quay - full support for running Quay deployments in air-gapped environments planned for future releases of Red Hat Quay

Red Hat



High-Availability

Red Hat

Quay High-Availability Setup

High-availability reference architecture prevents critical single PoF by running multiple instances of Quay.

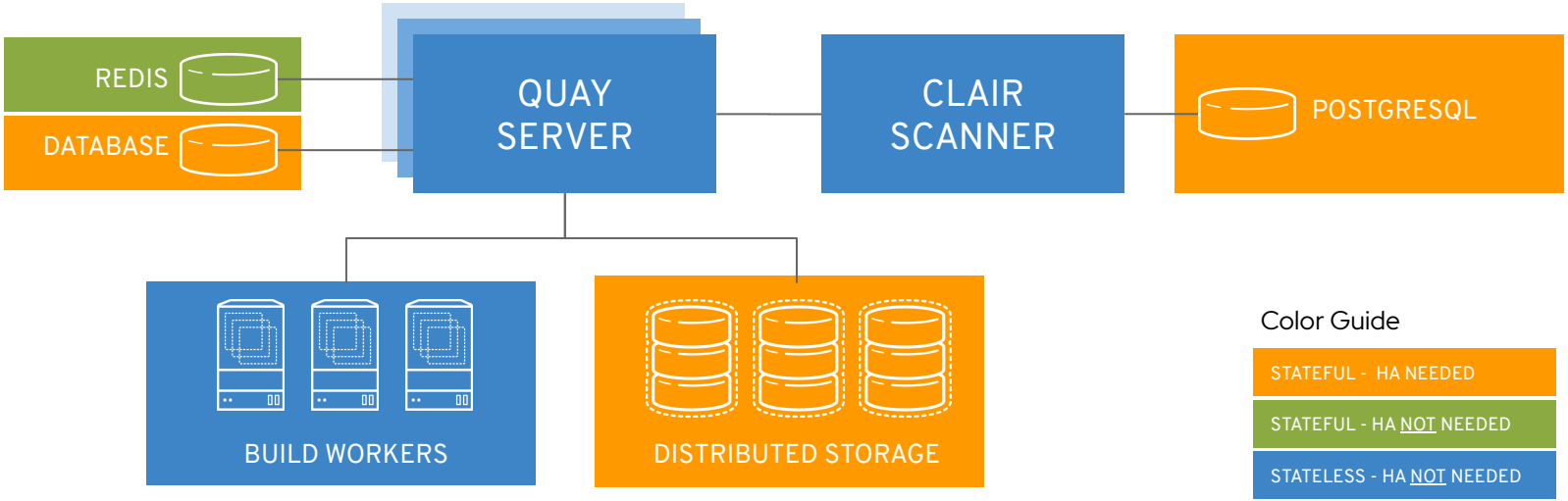
How it Works:

- Stateful components in HA mode
- Stateless components can be horizontally scaled arbitrarily

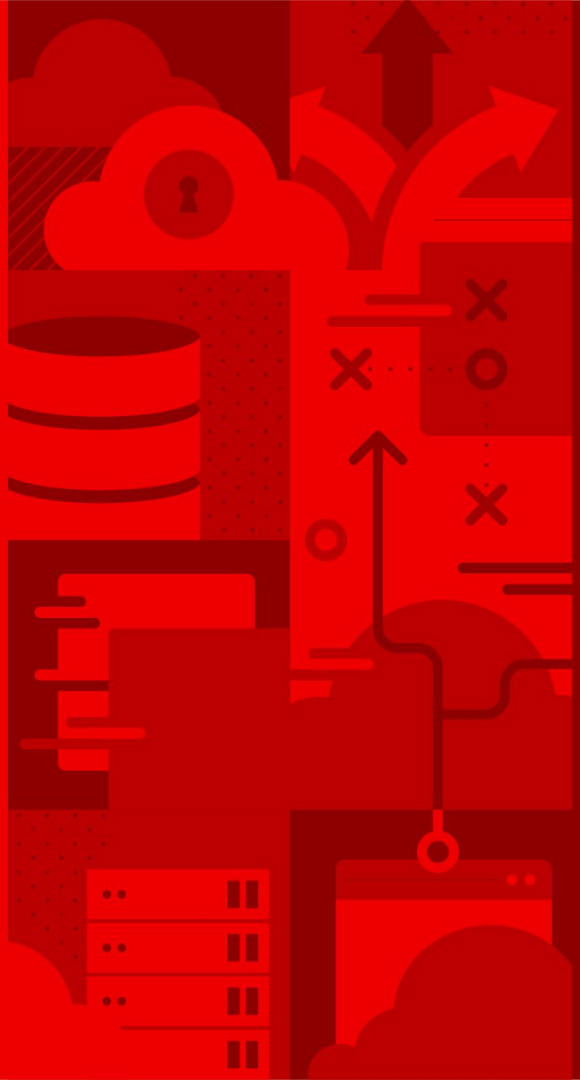
Note: Scaling out stateless components will add load to the stateful components, which must be accounted for in capacity planning.

https://access.redhat.com/documentation/en-us/red_hat_quay/3/html/deploy_red_hat_quay_-_high_availability/

Quay Components for HA



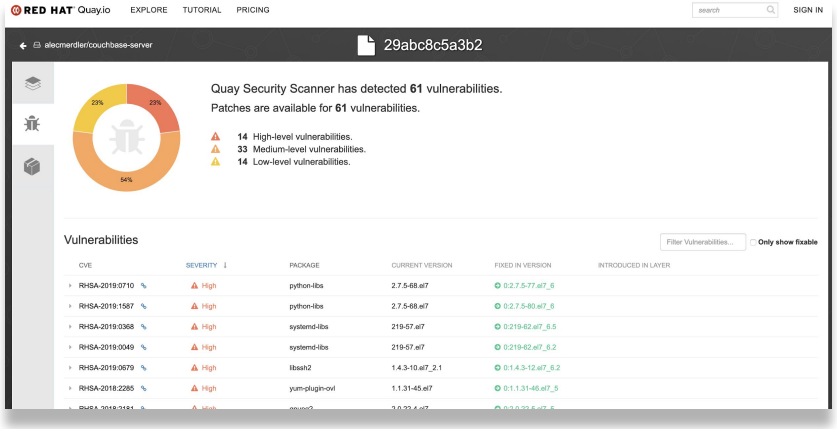
Red Hat



Built-In Vulnerability Scanning via Clair

Clair Overview

- Clair is an open source tool for **static analysis of vulnerabilities** in application containers
- Developed by CoreOS for Quay
- Used by various other projects and third party products
- Upstream Repository: https://github.com/quay/clair

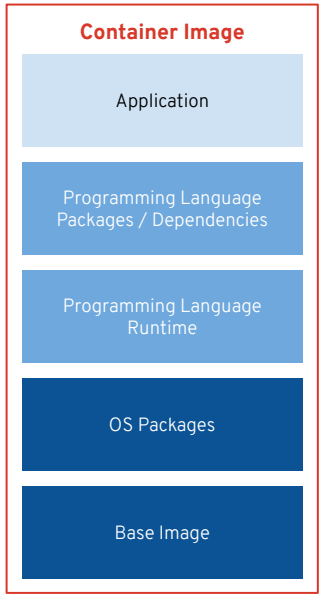


Clair - Supported Content Types

Clair v2 limited to OS level package managers including support for:

- Red Hat Enterprise Linux
- Debian Linux
- Ubuntu Linux
- Oracle Linux
- SuSE Linux
- Alpine Linux

Coming soon: Clair v4 (alongside Quay 3.4)



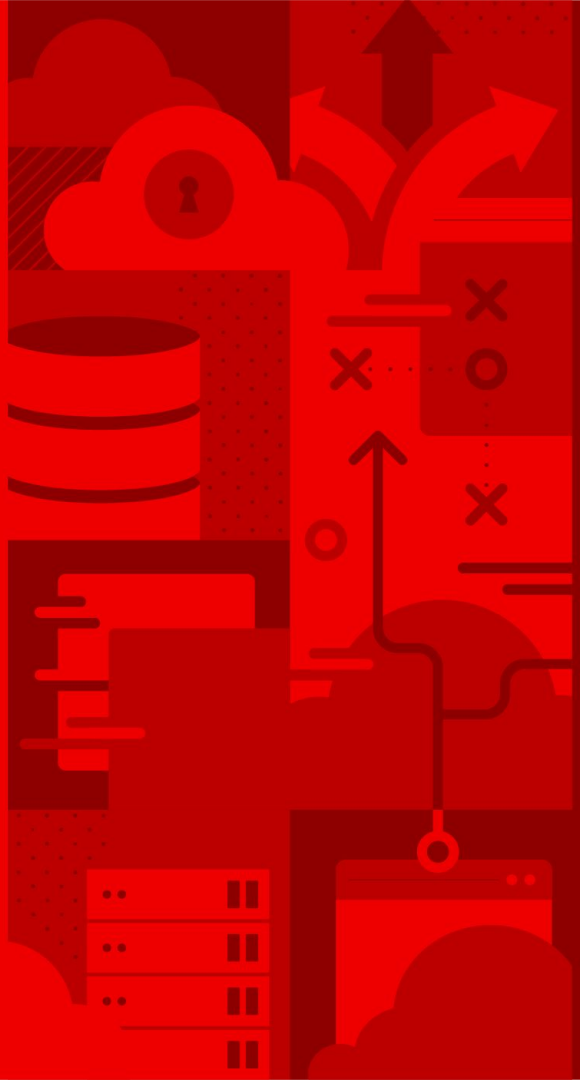
clair

Clair v4.0: Support for programing language package managers

Clair v4.x: Broader RH content coverage to cover non-RHEL RPM packages

Clair 2.9: support for various OS level package managers / distributions

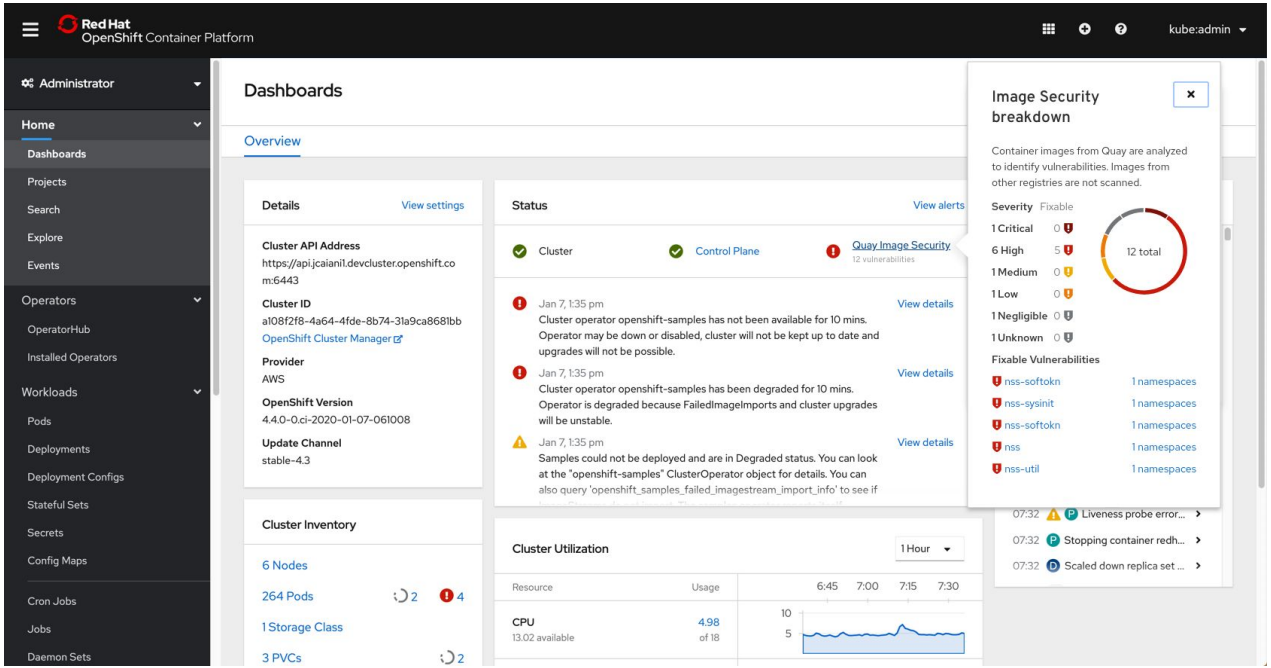
Red Hat



Container Security Operator and OpenShift Console Integration

Red Hat

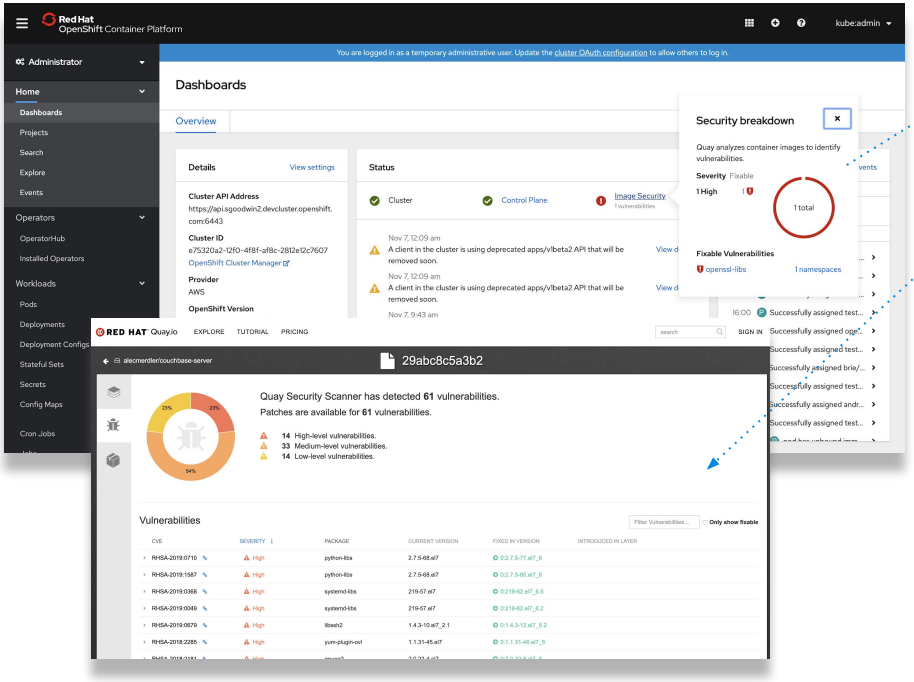
OpenShift Visualization of Vulnerabilities



- **Requires:**
  - Quay and Clair (on-cluster or off-cluster) and / or Quay.io used
  - CSO deployed and running on-cluster
- Container Security Operator (CSO) fetches vulnerability information from Clair
- Overview shown on OpenShift Cluster Admin Dashboard (Cluster Health Card)

Red Hat

Vulnerability Information in OpenShift Console



See all your Container Vulnerabilities right from the Console Dashboard

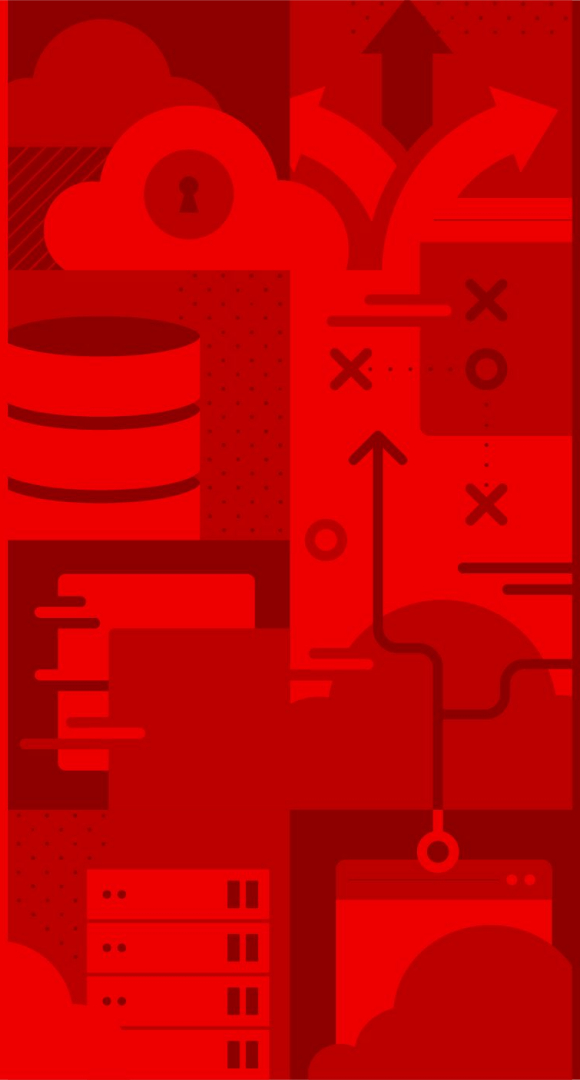
Link out to **Red Hat Quay** for more in depth information

Future Enhancements:

- data on a pod /project / cluster level
- Developer and admin console
- Reports incl. search filters / dashboards
- Notifications / Alerting

OpenShift Blog Post: https://blog.openshift.com/openshift-4-3-quay-container-security-integration/

Red Hat



Quay Roadmap

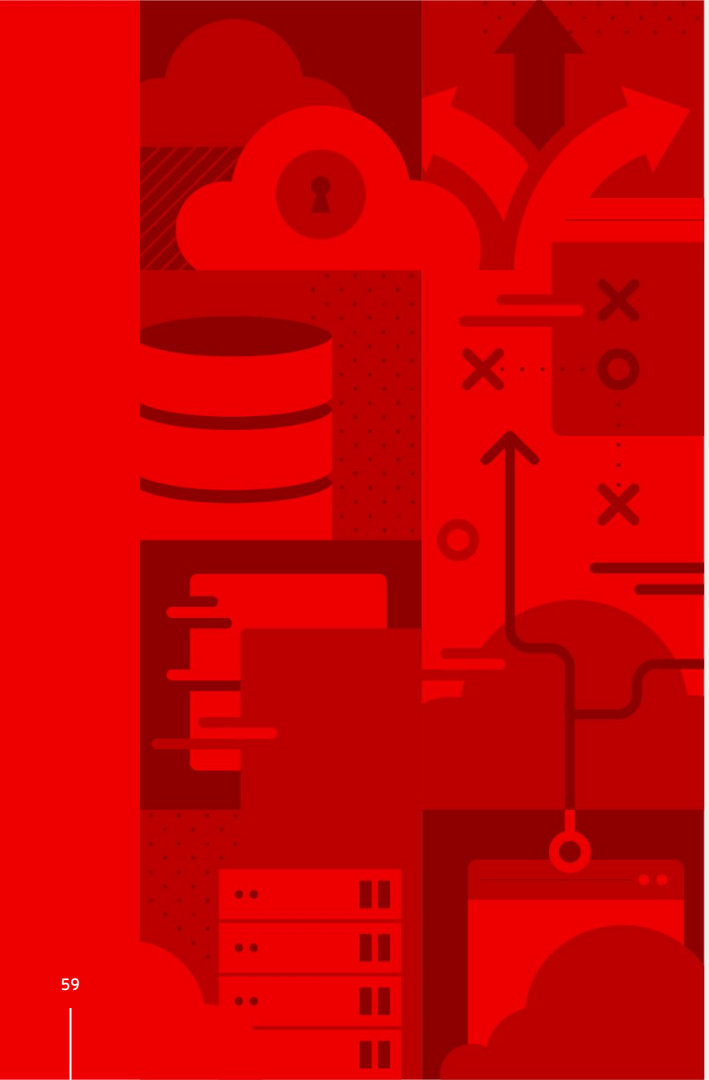
Red Hat

Red Hat Quay / Quay.io / Quay Dedicated Roadmap

	Near Term (3.3)	Mid Term (3.4+) *	Long Term (3.5+) *
QUAY	• Quay - OpenShift integration operator • LDAP filters / bindings • Usage (Quota) Reporting / Showback	• Setup operator enhancem. • CSO and OCP Console enh. • Repo mirroring enhancem. • Geo-Replication redesign • Quota Alerting / Enforcement	• Better air-gapped env support • Deeper integration into OCP • Quay Builder redesign • Content type Enhancements • App registry redesign
CLAIR	• **Clair v4 (Tech Preview):** • App content scanning • RH content accuracy • 1-2 prog languages	• Additional prog languages • Partner tooling integration • Clair air-gapped & multi-arch support	• Support for Microsoft Windows images • Enhanced content coverage • Container Health Index
HOSTED	• 2 SKUs for Quay.io (OF)	• Quay.io usage in OSD	• Quay Dedicated

Product Manager: Dirk Herrmann

* subject to change since release planning hasn't been done yet



Thank You!

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat