Chapter 3. Container Images

Abstract

Goal	Navigate container registries to find and manage container images.
Objectives	• Navigate container registries. • Pull and manage container images.
Sections	• Container Image Registries (and Guided Exercise) • Managing Images (and Guided Exercise)
Lab	• Container Images

Container Image Registries

Objectives

- Navigate container registries.

Container Registries

A container image is a packaged version of your application, with all the dependencies necessary for the application to run. You can use image registries to store container images to later share them in a controlled manner. Some examples of image registries include Quay.io, Red Hat Registry, Docker Hub, or Amazon ECR.

For example, consider the following Podman command.

```
[user@host ~]$ podman pull registry.redhat.io/ubi8/ubi:8.6
Trying to pull registry.redhat.io/ubi8/ubi:8.6...
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
3434...8f6b
```

The `registry.redhat.io/ubi8` image is stored in the Red Hat Registry, because the container name uses the `registry.redhat.io` host.

Red Hat Registry

Red Hat distributes container images by using two registries:

- `registry.access.redhat.com`: requires no authentication
- `registry.redhat.io`: requires authentication

However, Red Hat provides a centralized searching utility for both registries: the Red Hat Ecosystem Catalog, available at https://catalog.redhat.com/. You can use the Red Hat Ecosystem Catalog to search for images and get technical details about them. Go to https://catalog.redhat.com/software/containers/explore to search for container images.

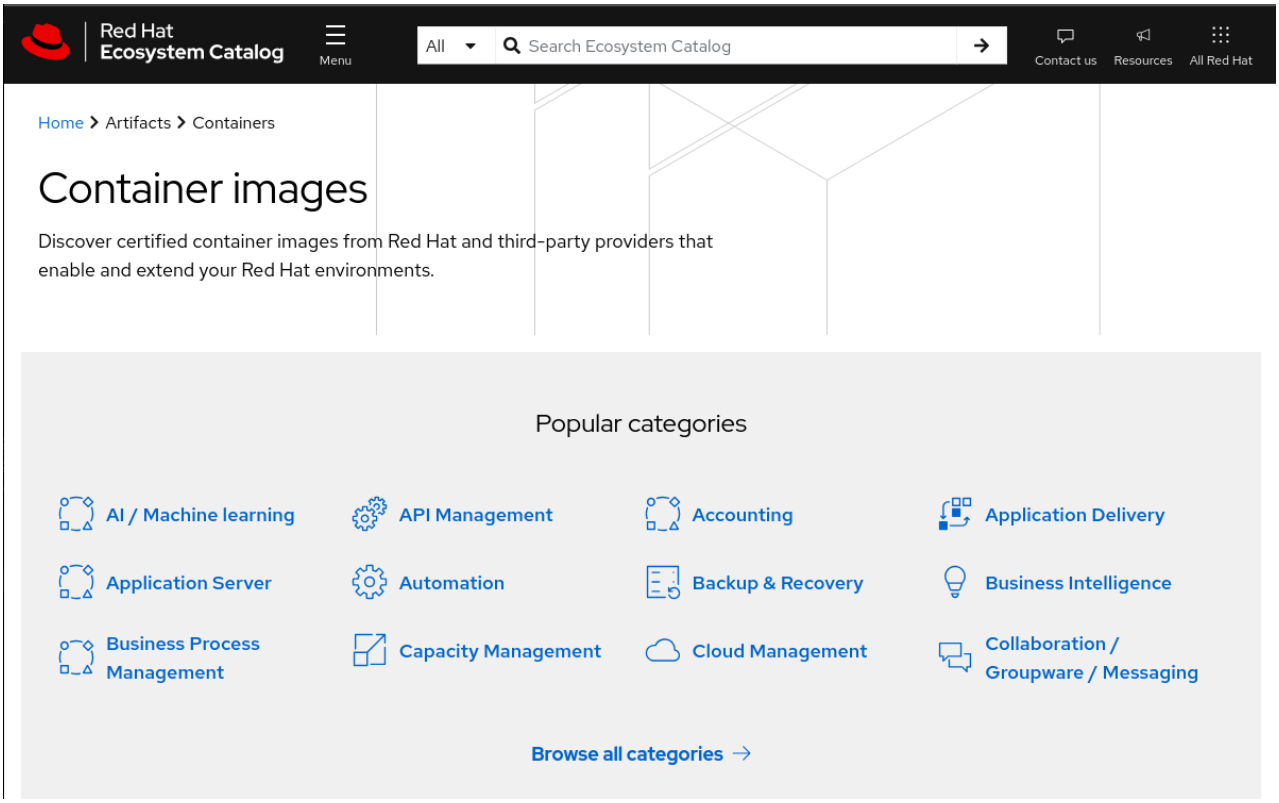

Figure 3.1: The Red Hat Ecosystem Catalog

The container image details page gives you relevant information about the container image, such as the Containerfile used to create the image, the packages installed within the image, or a security scanning. You can also change the image version by selecting a specific *tag*.

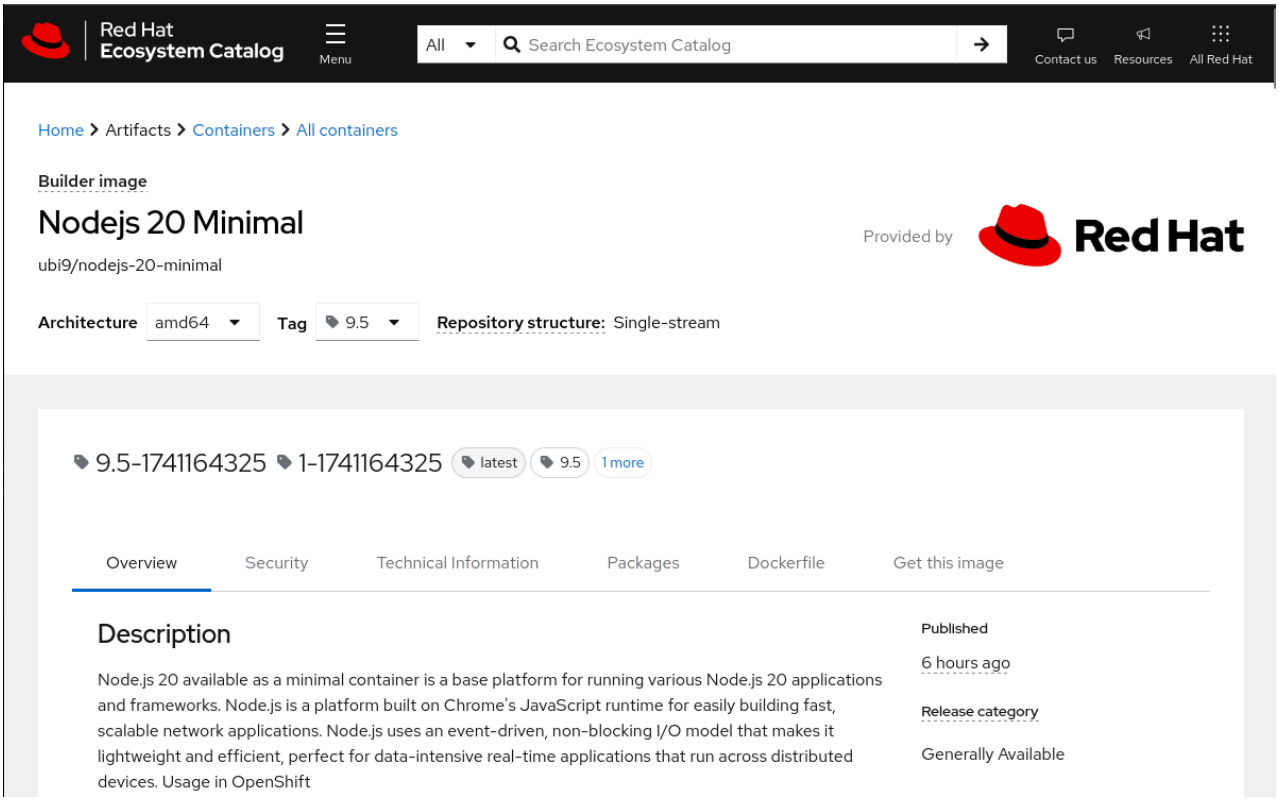

Figure 3.2: The Node.js 16 image based on RHEL 8

Useful Container Images

The following table introduces some useful container images.

Image	Provides	Description
`registry.access.redhat.com/ubi9`	Universal Base Image (UBI), Version 9	A base image to create other images that is based on RHEL 9.
`registry.access.redhat.com/ubi9/python-39`	Python 3.9	A UBI-based image with the Python 3.9 runtime.
`registry.access.redhat.com/ubi9/nodejs-18`	Node.js 18	A UBI-based image with the Node.js 18 runtime.
`registry.access.redhat.com/ubi9/go-toolset`	Go Toolset	A UBI-based image with the Go runtime. The Go version depends on the image tag.

Red Hat UBIs are *Open Container Initiative (OCI)* compliant enterprise grade container images that provide the base operating system layer for your containerized applications. UBIs include a subset of Red Hat Enterprise Linux (RHEL) components. UBIs can additionally provide a set of pre-built language runtimes. UBIs are freely distributable, and you can use UBIs on both Red Hat and non-Red Hat platforms or container registries.

You do not need a Red Hat subscription to use or distribute UBI-based images. However, Red Hat only provides full support for containers built on UBI if the containers are deployed to a Red Hat platform, such as Red Hat OpenShift Container Platform (RHOCP) or RHEL.

Quay.io

Although the Red Hat Registry only stores images from Red Hat and certified providers, you can use the Quay.io registry to store your custom images. Storing public images in Quay.io is free, and paying customers receive further benefits, such as private repositories. Developers can also deploy an on-premise Quay instance, which you can use to set up an image registry on your infrastructure.

To log in to Quay.io, you can use your Red Hat developer account.

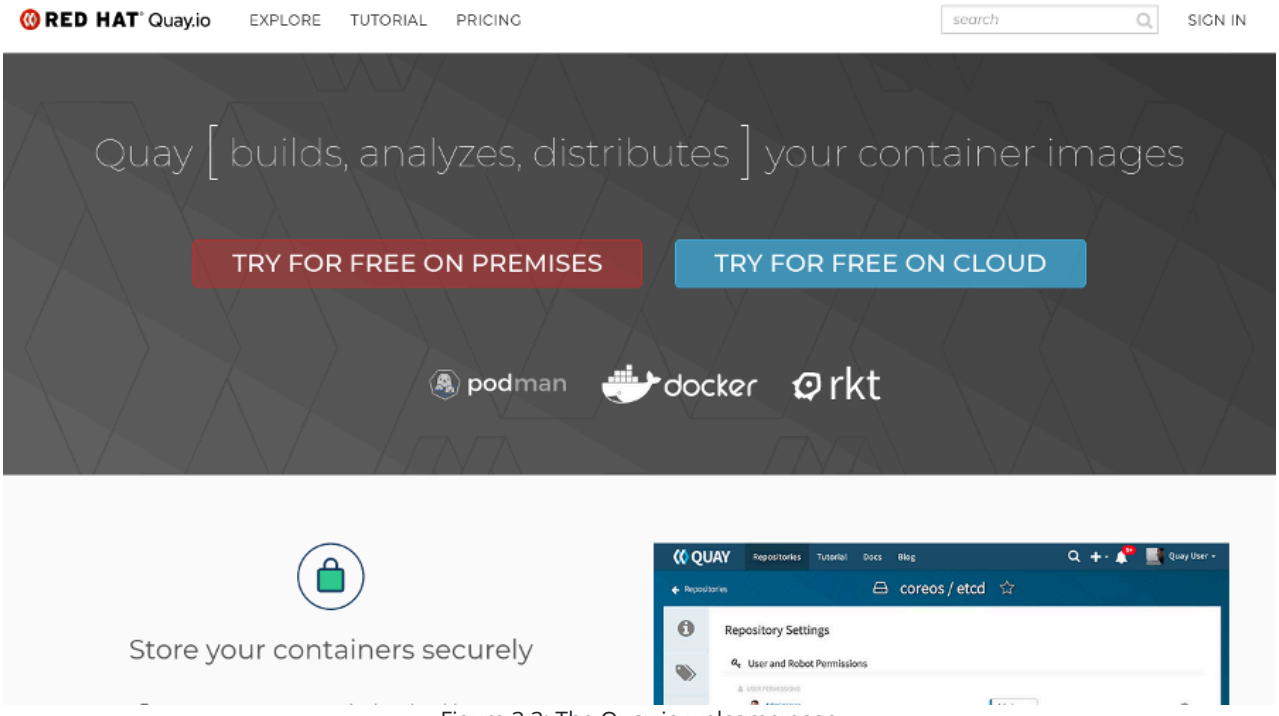

Figure 3.3: The Quay.io welcome page

Manage Registries with Podman

When you pull a container image, you provide a number of details. For example, the `registry.access.redhat.com/ubi9/nodejs-18:latest` image name consists of the following information:

- Registry URL: `registry.access.redhat.com`
- User or organization: `ubi9`
- Image repository: `nodejs-18`
- Image tag: `latest`

Developers can specify a shorter, unqualified name, which omits the registry URL. For example, you might shorten the `registry.redhat.io/rhel7/rhel:7.9` to `rhel7/rhel:7.9`.

```
[user@host ~]$ podman pull ubi8/python-39
```

If you do not provide the registry URL, then Podman uses the `~/.config/containers/registries.conf` or `/etc/containers/registries.conf` files to search other container registries that might contain the image name. These files contains registries that Podman searches to find the image, in order of preference. The user-defined `~/.config/containers/registries.conf` file overrides the global `/etc/containers/registries.conf` file.

For example, with the following configuration, Podman searches the Red Hat Registry first. If the image is not found in the Red Hat Registry, then Podman searches in the Docker Hub registry.

```
unqualified-search-registries == ['registry.redhat.io', 'docker.io']
```

You can also block a registry. For example, the following configuration blocks pulling from the Docker Hub.

```
[[registry]]
location="docker.io"
blocked=true
```

Some systems do not have a `/etc/containers/registries.conf` file, such as Microsoft Windows. In such cases, the `registries.conf` file might exist in a different location.

For example, on Microsoft Windows, execute `podman machine ssh` to connect to the Linux-based virtual machine that starts your containers. In the virtual machine, you can find the `/etc/containers/registries.conf` file:

```
[user@host ~]$ podman machine ssh
[user@DESKTOP-AA1A111 ~]$ ls /etc/containers/containers.conf
/etc/containers/containers.conf
```

See the references section for more details about the `registries.conf` file.

Red Hat recommends that you always use a fully qualified container image name to avoid using images from unintended container registries. For example, depending on your Podman configuration, the `rhel7/rhel:7.9` container image might resolve to a potentially unsupported or malicious `docker.io/library/rhel7/rhel:7:9` container image.

If Podman matches the short image name in several registries, then it prompts the user to select which to use.

```
[user@host ~]$ podman pull httdp
? Please select an image:
  ▸ registry.access.redhat.com/httdp:latest
    registry.redhat.io/httdp:latest
    docker.io/library/httdp:latest
```

Manage Registries with Skopeo

Skopeo is a command-line tool for working with container images. Developers can use Skopeo in a number of ways, for example:

- Inspect remote container images.
- Copy a container image between registries.
- Sign an image with OpenPGP keys.
- Convert image format, for example from `Docker` to the `OCI` format.

Skopeo can inspect remote images or transfer images between registries without using local storage. The `skopeo` command uses the `transport:image` format, such as `docker://remote_image`, `dir:path`, or `oci:path:tag`.

Use the `skopeo inspect` command to read image metadata.

```
[user@host ~]$ skopeo inspect \
 docker://registry.access.redhat.com/ubi9/nodejs-18
{
    "Name": "registry.access.redhat.com/ubi9/nodejs-18",
    "Digest": "sha256:741b...22e0",
    "RepoTags": [
...output omitted...
```

Use the `skopeo copy` command to copy images between registries. The following example copies the `registry.access.redhat.com/ubi9/nodejs-18:latest` image into the `quay.io/myuser/nodejs-18` Quay.io repository.

```
[user@host ~]$ skopeo copy \
 docker://registry.access.redhat.com/ubi9/nodejs-18 \
 docker://quay.io/myuser/nodejs-18
Getting image source signatures
...output omitted...
```

The following example changes the transport format to download an image into a local directory.

```
[user@host ~]$ skopeo copy \
 docker://registry.access.redhat.com/ubi9/nodejs-18 \
 dir:/var/lib/images/nodejs-18
Getting image source signatures
...output omitted...
```

See the references section for more details about Skopeo.

Manage Registry Credentials with Podman

Some registries require users to authenticate, such as the `registry.redhat.io` registry.

```
[user@host ~]$ podman pull registry.redhat.io/rhel8/httpd-24
Trying to pull registry.redhat.io/rhel8/httpd-24:latest...
Error: initializing source docker://registry.redhat.io/rhel8/httpd-24:latest: unable to retrieve auth token: invalid
username/password: unauthorized: Please login to the Red Hat Registry using your Customer Portal credentials. Further instruc
tions can be found here: https://access.redhat.com/RegistryAuthentication
```

You might choose a different image that does not require authentication, such as the UBI 8 image from the `registry.access.redhat.com` registry:

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8:latest
Trying to pull registry.access.redhat.com/ubi8:latest...
Getting image source signatures
Checking if image destination supports signatures
...output omitted...
```

Alternatively, authenticate your calls by executing the `podman login` command.

```
[user@host ~]$ podman login registry.redhat.io
Username: YOUR_USER
Password: YOUR_PASSWORD
Login Succeeded!
[user@host ~]$ podman pull registry.redhat.io/rhel8/httpd-24
Trying to pull registry.redhat.io/rhel8/httpd-24:latest...
Getting image source signatures
...output omitted...
```

Podman stores the credentials in the `${XDG_RUNTIME_DIR}/containers/auth.json` file, where the `${XDG_RUNTIME_DIR}` refers to a directory specific to the current user. The credentials are encoded in the `base64` format:

```
[user@host ~]$ cat ${XDG_RUNTIME_DIR}/containers/auth.json
{
        "auths": {
                "registry.redhat.io": {
                        "auth": "dXNlcjpodW50ZXIy"
                }
        }
}
[user@host ~]$ echo -n dXNlcjpodW50ZXIy | base64 -d
user:hunter2
```

> **NOTE**
>
> For security reasons, the `podman login` command does not show your password in the interactive session. Although you do not see what you are typing, Podman registers every key stroke. Press enter when you have typed your full password in the interactive session to initiate the login.

Skopeo uses the same `${XDG_RUNTIME_DIR}/containers/auth.json` file to access authentication details for each registry.

REFERENCES

Red Hat - Manage Container Registries

`skopeo(1)` man page

`skopeo-copy(1)` man page