Rocky Enterprise Linux 9.2 Manual Pages on command 'containers-registries.d.5'

$ man containers-registries.d.5

containers-registries.d(5) Page containers-registries.d(5)

Miloslav Trma? August 2016

NAME

containers-registries.d - Directory for various registries configura?

tions

DESCRIPTION

The registries configuration directory contains configuration for vari?

ous registries (servers storing remote container images), and for con?

tent stored in them, so that the configuration does not have to be pro?

vided in command-line options over and over for every command, and so

that it can be shared by all users of containers/image.

By default, the registries configuration directory is $HOME/.con?

fig/containers/registries.d if it exists, otherwise /etc/contain?

ers/registries.d (unless overridden at compile-time); applications may

allow using a different directory instead.

Directory Structure

The directory may contain any number of files with the extension .yaml,

each using the YAML format. Other than the mandatory extension, names

of the files don?t matter.

The contents of these files are merged together; to have a well-defined and easy to understand behavior, there can be only one configuration section describing a single namespace within a registry (in particular there can be at most one one default-docker section across all files, and there can be at most one instance of any key under the docker sec? tion; these sections are documented later).

Thus, it is forbidden to have two conflicting configurations for a sin? gle registry or scope, and it is also forbidden to split a configura? tion for a single registry or scope across more than one file (even if they are not semantically in conflict).

Registries, Scopes and Search Order

Each YAML file must contain a ?YAML mapping? (key-value pairs). Two top-level keys are defined:

? default-docker is the configuration section (as documented be? low) for registries implementing "Docker Registry HTTP API V2".

This key is optional.

? docker is a mapping, using individual registries implementing "Docker Registry HTTP API V2", or namespaces and individual images within these registries, as keys; the value assigned to any such key is a configuration section.

This key is optional.

Scopes matching individual images are named Docker references in the fully expanded form, either

using a tag or digest. For example, docker.io/library/busybox:latest (not busybox:latest).

More general scopes are prefixes of individual-image scopes, and spec? ify a repository (by omitting the tag or digest),

a repository namespace, or a registry host (and a port if it differs from the default).

Note that if a registry is accessed using a hostname+port configura? tion, the port-less hostname

is not used as parent scope.

When searching for a configuration to apply for an individual container image, only the configuration for the most-precisely matching scope is used; configuration using more general scopes is ignored. For example, if any configuration exists for docker.io/library/busybox, the configu? ration for docker.io is ignored (even if some element of the configura? tion is defined for docker.io and not for docker.io/library/busybox).

Built-in Defaults

If no docker section can be found for the container image, and no de? fault-docker section is configured:

? The default directory, /var/lib/containers/sigstore for root and $HOME/.local/share/containers/sigstore for unprivileged user, will be used for reading and writing signatures.

? Sigstore attachments will not be read/written.

Individual Configuration Sections

A single configuration section is selected for a container image using the process described above. The configuration section is a YAML map? ping, with the following keys:

? lookaside-staging defines an URL of of the signature storage, used for editing it (adding or deleting signatures).

This key is optional; if it is missing, lookaside below is used.

? lookaside defines an URL of the signature storage. This URL is used for reading existing signatures, and if lookaside-staging does not exist, also for adding or removing them.

This key is optional; if it is missing, no signature storage is defined (no signatures

are download along with images, adding new signatures is possible only if lookaside-staging is defined).

? use-sigstore-attachments specifies whether sigstore image at? tachments (signatures, attestations and the like) are going to be read/written along with the image. If disabled, the images are treated as if no attachments exist; attempts to write at? tachments fail.

Examples

Using Containers from Various Origins

The following demonstrates how to to consume and run images from vari?

ous registries and namespaces:

```
docker:
  registry.database-supplier.com:
    lookaside: https://lookaside.database-supplier.com
  distribution.great-middleware.org:
    lookaside: https://security-team.great-middleware.org/lookaside
  docker.io/web-framework:
    lookaside: https://lookaside.web-framework.io:8080
```

Developing and Signing Containers, Staging Signatures

For developers in example.com:

? Consume most container images using the public servers also used by clients.

? Use a separate signature storage for an container images in a namespace corresponding to the developers' department, with a staging storage used before publishing signatures.

? Craft an individual exception for a single branch a specific developer is working on locally.

```
docker:
  registry.example.com:
    lookaside: https://registry-lookaside.example.com
  registry.example.com/mydepartment:
    lookaside: https://lookaside.mydepartment.example.com
    lookaside-staging: file:///mnt/mydepartment/lookaside-staging
  registry.example.com/mydepartment/myproject:mybranch:
    lookaside: http://localhost:4242/lookaside
    lookaside-staging: file:///home/useraccount/webroot/lookaside
```

A Global Default

If a company publishes its products using a different domain, and dif?

ferent registry hostname for each of them, it is still possible to use a single signature storage server without listing each domain individu?

ally. This is expected to rarely happen, usually only for staging new signatures.

default-docker:

lookaside-staging: file:///mnt/company/common-lookaside-staging

AUTHORS

Miloslav Trma? mitr@redhat.com ?mailto:mitr@redhat.com?