Red Hat Enterprise Linux Release 9.2 Manual Pages on 'containers-registries.conf.5' command

$ man containers-registries.conf.5

CONTAINERS-REGISTRIES.CONF(5)   configuration   CONTAINERS-REGISTRIES.CONF(5)

Brent Baude Aug 2017

NAME

containers-registries.conf - Syntax of System Registry Configuration

File

DESCRIPTION

The CONTAINERS-REGISTRIES configuration file is a system-wide configu?

ration file for container image registries. The file format is TOML.

Container engines will use the $HOME/.config/containers/registries.conf

if it exists, otherwise they will use /etc/containers/registries.conf

GLOBAL SETTINGS

unqualified-search-registries

An array of host[:port] registries to try when pulling an un?

qualified image, in order.

credential-helpers

An array of default credential helpers used as external creden?

tial stores. Note that "containers-auth.json" is a reserved

value to use auth files as specified in containers-auth.json(5).

The credential helpers are set to ["containers-auth.json"] if

none are specified.

NAMESPACED [[registry]] SETTINGS

The bulk of the configuration is represented as an array of [[reg?

istry]] TOML tables; the settings may therefore differ among different registries as well as among different namespaces/repositories within a registry.

Choosing a [[registry]] TOML table

Given an image name, a single [[registry]] TOML table is chosen based on its prefix field.

prefix: A prefix of the user-specified image name, i.e. using one of the following formats:

- host[:port]
- host[:port]/namespace[/_namespace_?]
- host[:port]/namespace[/_namespace_?]/repo
- host[:port]/namespace[/_namespace_?]/repo(:_tag|@digest)
- [*.]host

The user-specified image name must start with the specified prefix (and continue with the appropriate separator) for a particular [[registry]] TOML table to be considered; (only) the TOML table with the longest match is used. It can also include wildcarded subdomains in the format *.example.com. The wildcard should only be present at the beginning as shown in the formats above. Other cases will not work. For example, *.example.com is valid but example.*.com, *.example.com/foo and *.exam?ple.com:5000/foo/bar:baz are not. Note that * matches an arbitrary number of subdomains. *.example.com will hence match bar.example.com, foo.bar.example.com and so on.

As a special case, the prefix field can be missing; if so, it defaults to the value of the location field (described below).

Per-namespace settings

insecure

true or false. By default, container runtimes require TLS when retrieving images from a registry. If insecure is set to true, unencrypted HTTP as well as TLS connections with untrusted cer?tificates are allowed.

blocked

true or false. If true, pulling images with matching names is forbidden.

Remapping and mirroring registries

The user-specified image reference is, primarily, a "logical" image name, always used for naming the image. By default, the image refer? ence also directly specifies the registry and repository to use, but the following options can be used to redirect the underlying accesses to different registry servers or locations (e.g. to support configura? tions with no access to the internet without having to change Docker? files, or to add redundancy).

location

Accepts the same format as the prefix field, and specifies the physical location of the prefix-rooted namespace.

By default, this equal to prefix (in which case prefix can be omitted and the [[registry]] TOML table can only specify location).

Example: Given

```
prefix = "example.com/foo"
location = "internal-registry-for-example.net/bar"
```

requests for the image example.com/foo/myimage:latest will actually work with the internal-registry-for-example.net/bar/myimage:latest im? age.

With a prefix containing a wildcard in the format: "*.example.com" for subdomain matching, the location can be empty. In such a case, prefix matching will occur, but no reference rewrite will occur. The original requested image string will be used as-is. But other settings like in? secure / blocked / mirrors will be applied to matching images.

Example: Given

```
prefix = "*.example.com"
```

requests for the image blah.example.com/foo/myimage:latest will be used as-is. But other settings like insecure/blocked/mirrors will be applied to matching images

mirror An array of TOML tables specifying (possibly-partial) mirrors for the prefix-rooted namespace (i.e., the current [[registry]] TOML table).

The mirrors are attempted in the specified order; the first one that can be contacted and contains the image will be used (and if none of the mirrors contains the image, the primary location specified by the registry.location field, or using the unmodified user-specified refer? ence, is tried last).

Each TOML table in the mirror array can contain the following fields: - location? same semantics as specified in the [[registry]] TOML table - insecure? same semantics as specified in the [[registry]] TOML table - pull-from-mirror: all, digest-only or tag-only. If "digest-only"? mirrors will only be used for digest pulls. Pulling images by tag can potentially yield different images, depending on which endpoint we pull from. Restricting mirrors to pulls by digest avoids that issue. If "tag-only", mirrors will only be used for tag pulls. For a more up-to-date and expensive mirror that it is less likely to be out of sync if tags move, it should not be unnecessarily used for digest references. Default is "all" (or left empty), mirrors will be used for both digest pulls and tag pulls unless the mirror-by-digest-only is set for the primary registry. Note that this per-mirror setting is allowed only when mirror-by-digest-only is not configured for the primary registry.

mirror-by-digest-only

true or false. If true, mirrors will only be used during pulling if the image reference includes a digest. Note that if all mirrors are configured to be digest-only, images referenced by a tag will only use the primary registry. If all mirrors are configured to be tag-only, images referenced by a digest will only use the primary registry.

Referencing an image by digest ensures that the same is always used (whereas referencing an image by a tag may cause different registries to return different images if the tag mapping is out of sync).

Note: Redirection and mirrors are currently processed only when reading images, not when pushing to a registry; that may change in the future.

Short-Name Aliasing

The use of unqualified-search registries entails an ambiguity as it is unclear from which registry a given image, referenced by a short name, may be pulled from.

As mentioned in the note at the end of this man page, using short names is subject to the risk of hitting squatted registry namespaces. If the unqualified-search registries are set to ["registry1.com", "reg?
istry2.com"] an attacker may take over a namespace of registry1.com such that an image may be pulled from registry1.com instead of the in?
tended source registry2.com.

While it is highly recommended to always use fully-qualified image ref?
erences, existing deployments using short names may not be easily changed. To circumvent the aforementioned ambiguity, so called short-name aliases can be configured that point to a fully-qualified image reference.

Short-name aliases can be configured in the [aliases] table in the form of "name"="value" with the left-hand name being the short name (e.g., "image") and the right-hand value being the fully-qualified image ref?
erence   (e.g., "registry.com/namespace/image").   Note that neither "name" nor "value" can include a tag or digest. Moreover, "name" must be a short name and hence cannot include a registry domain or refer to localhost.

When pulling a short name, the configured aliases table will be used for resolving the short name. If a matching alias is found, it will be used without further consulting the unqualified-search registries list.

If no matching alias is found, the behavior can be controlled via the short-name-mode option as described below.

Note that tags and digests are stripped off a user-specified short name for alias resolution. Hence, "image", "image:tag" and "image@digest" all resolve to the same alias (i.e., "image"). Stripped off tags and digests are later appended to the resolved alias.

Further note that drop-in configuration files (see containers-reg?
istries.conf.d(5)) can override aliases in the specific loading order of the files.   If the "value" of an alias is empty (i.e., ""), the alias will be erased. However, a given "name" may only be specified once in a single config file.

Short-Name Aliasing: Modes

The short-name-mode option supports three modes to control the behav?
iour of short-name resolution.

- ? enforcing: If only one unqualified-search registry is set, use it as there is no ambiguity. If there is more than one reg? istry and the user program is running in a terminal (i.e., stdout & stdin are a TTY), prompt the user to select one of the specified search registries. If the program is not run? ning in a terminal, the ambiguity cannot be resolved which will lead to an error.
- ? permissive: Behaves as enforcing but does not lead to an error if the program is not running in a terminal. Instead, fall? back to using all unqualified-search registries.
- ? disabled: Use all unqualified-search registries without prompting.

If short-name-mode is not specified at all or left empty, default to the permissive mode. If the user-specified short name was not aliased already, the enforcing and permissive mode if prompted, will record a new alias after a successful pull. Note that the recorded alias will be written to /var/cache/containers/short-name-aliases.conf for root to have a clear separation between possibly human-edited registries.conf files and the machine-generated short-name-aliases-conf. Note that $HOME/.cache is used for rootless users. If an alias is specified in a registries.conf file and also the machine-generated short-name-aliases.conf, the short-name-aliases.conf file has precedence.

Normalization of docker.io references

The Docker Hub docker.io is handled in a special way: every push and pull operation gets internally normalized with /library if no other specific namespace is defined (for example on docker.io/namespace/im? age).

(Note that the above-described normalization happens to match the be? havior of Docker.)

This means that a pull of docker.io/alpine will be internally trans?

lated to docker.io/library/alpine. A pull of docker.io/user/alpine will not be rewritten because this is already the correct remote path. Therefore, to remap or mirror the docker.io images in the (implied) /library namespace (or that whole namespace), the prefix and location fields in this configuration file must explicitly include that /library namespace. For example prefix = "docker.io/library/alpine" and not pre? fix = "docker.io/alpine". The latter would match the docker.io/alpine/* repositories but not the docker.io/[library/]alpine image).

EXAMPLE

```
unqualified-search-registries = ["example.com"]

[[registry]]

prefix = "example.com/foo"

insecure = false

blocked = false

location = "internal-registry-for-example.com/bar"

[[registry.mirror]]

location = "example-mirror-0.local/mirror-for-foo"

[[registry.mirror]]

location = "example-mirror-1.local/mirrors/foo"

insecure = true

[[registry]]

location = "registry.com"

[[registry.mirror]]

location = "mirror.registry.com"
```

Given the above, a pull of example.com/foo/image:latest will try:

1. example-mirror-0.local/mirror-for-foo/image:latest
2. example-mirror-1.local/mirrors/foo/image:latest
3. internal-registry-for-example.net/bar/image:latest

in order, and use the first one that exists.

Note that a mirror is associated only with the current [[registry]] TOML table. If using the example above, pulling the image reg? istry.com/image:latest will hence only reach out to mirror.reg? istry.com, and the mirrors associated with example.com/foo will not be considered.

VERSION 1 FORMAT - DEPRECATED

VERSION 1 format is still supported but it does not support using reg?
istry mirrors, longest-prefix matches, or location rewriting.

The TOML format is used to build a simple list of registries under
three categories: registries.search, registries.insecure, and reg?
istries.block. You can list multiple registries using a comma sepa?
rated list.

Search registries are used when the caller of a container runtime does
not fully specify the container image that they want to execute. These
registries are prepended onto the front of the specified container im?
age until the named image is found at a registry.

Note that insecure registries can be used for any registry, not just
the registries listed under search.

The registries.insecure and registries.block lists have the same mean?
ing as the insecure and blocked fields in the current version.

EXAMPLE

The following example configuration defines two searchable registries,
one insecure registry, and two blocked registries.

```
[registries.search]
registries = ['registry1.com', 'registry2.com']
[registries.insecure]
registries = ['registry3.com']
[registries.block]
registries = ['registry.untrusted.com', 'registry.unsafe.com']
```

NOTE: RISK OF USING UNQUALIFIED IMAGE NAMES

We recommend always using fully qualified image names including the
registry server (full dns name), namespace, image name, and tag (e.g.,
registry.redhat.io/ubi8/ubi:latest). When using short names, there is
always an inherent risk that the image being pulled could be spoofed.
For example, a user wants to pull an image named foobar from a registry
and expects it to come from myregistry.com. If myregistry.com is not
first in the search list, an attacker could place a different foobar

image at a registry earlier in the search list. The user would acciden?
tally pull and run the attacker's image and code rather than the in?
tended content. We recommend only adding registries which are com?
pletely trusted, i.e. registries which don't allow unknown or anonymous
users to create accounts with arbitrary names. This will prevent an im?
age from being spoofed, squatted or otherwise made insecure. If it is
necessary to use one of these registries, it should be added at the end
of the list.

It is recommended to use fully-qualified images for pulling as the des?
tination registry is unambiguous. Pulling by digest (i.e.,
quay.io/repository/name@digest) further eliminates the ambiguity of
tags.

SEE ALSO

containers-auth.json(5) containers-certs.d(5)

HISTORY

Dec 2019, Warning added for unqualified image names by Tom Sweeney
tsweeney@redhat.com ?mailto:tsweeney@redhat.com?

Mar 2019, Added additional configuration format by Sascha Grunert
sgrunert@suse.com ?mailto:sgrunert@suse.com?

Aug 2018, Renamed to containers-registries.conf(5) by Valentin Rothberg
vrothberg@suse.com ?mailto:vrothberg@suse.com?

Jun 2018, Updated by Tom Sweeney tsweeney@redhat.com
?mailto:tsweeney@redhat.com?

Aug 2017, Originally compiled by Brent Baude bbaude@redhat.com
?mailto:bbaude@redhat.com?