

Image Registries

Glyn Normington

May 16, 2019

This document provides a formal model of image registries.

Contents

1	Introduction	1
2	Overview of this document	1
3	Fundamentals	1
4	Content Digests	2
5	Repositories	3
6	Registries	5
7	Image References	7
A	Z Notation	9

1 Introduction

This document provides a formal model of image registries.

2 Overview of this document

Docker Inc. introduced container images and registries to hold them and these were later standardised as part of the Open Container Initiative.

This document models image references, repositories, and registries. It covers digests and tags.

The Z specification language is used to capture the model, but sufficient English text is also provided that readers who do not know Z should be able to understand the model. The appendix contains a summary of the Z notation. For more information about Z, please consult the Z Manual (<https://www.cs.umd.edu/~mvz/handouts/z-manual.pdf>). The model was type checked using fuzz (<https://bitbucket.org/Spivey/fuzz>).

3 Fundamentals

Images are opaque blobs as far as we are concerned here - the decomposition into layers is ignored. Similarly, cryptographic hashes, or *hexes* to use the terminology of the OCI Distribution specification, tags, and (registry) hostnames and paths are modelled, but their details are not.

$[Image, Hex, Tag, Hostname, Path]$

There is a special reserved tag.

| *Latest* : *Tag*

4 Content Digests

A content digest is a combination of a cryptographic hash function (such as SHA-256) or "algorithm, and the hash output by such a function.

$\begin{array}{l} \textit{ContentDigest} \\ \textit{alg} : \textit{Image} \rightarrow \textit{Hex} \\ \textit{hash} : \textit{Hex} \end{array}$

The idea is that a content digest d identifies an image i if and only if:

$$d.\textit{alg} \ i = d.\textit{hash}$$

An optional content digest is modelled as a datatype.

$$\textit{OptionalContentDigest} ::= \textit{None} \mid \textit{Dig}\langle\langle \textit{ContentDigest} \rangle\rangle$$

5 Repositories

A repository is a collection of images indexed by content digest and by tag.

<i>Repo</i>	_____
$cd : ContentDigest \rightarrow Image$	
$tag : Tag \rightarrow Image$	
$\forall d : \text{dom } cd \bullet d.alg(cd\ d) = d.hash$	
$\text{ran } tag \subseteq \text{ran } cd$	

The content digests identify the corresponding images. Each image identified by a tag is also identified by a content digest.

Initially, a repository is empty.

<i>RepoInit</i>	_____
<i>Repo'</i>	
$cd' = \emptyset$	
$tag' = \emptyset$	

An image is added to a repository by *pushing* it.

<i>RepoPush</i>	_____
$\Delta Repo$	
$i? : Image$	
$d? : ContentDigest$	
$t? : Tag$	
$cd' = cd \oplus \{d? \mapsto i?\}$	
$tag' = tag \oplus \{t? \mapsto i?\}$	

An image is retrieved from a repository by *pulling* it.

We can either pull using a content digest

<i>RepoPullByDigest</i>	_____
$\Xi Repo$	
$d? : OptionalContentDigest$	
$i! : Image$	
$\exists d : ContentDigest \mid d? = Dig\ d \wedge d \in \text{dom } cd \bullet i! = cd\ d$	

or, if a content digest is not supplied, by using a tag.

$RepoPullByTag$	_____
$\exists Repo$	
$t? : Tag$	
$d? : OptionalContentDigest$	
$i! : Image$	
$d? = None$	
$t? \in \text{dom } tag \Rightarrow i! = tag\ t?$	

$$RepoPullOk \hat{=} RepoPullByDigest \vee RepoPullByTag$$

6 Registries

A registry is a collection of repositories index by path.

$$\frac{\text{Registry}}{repo : Path \rightarrow Repo}$$

Paths which do not exist are modelled as pointing to empty repositories.

Initially a registry has only empty repositories.

$$\frac{\frac{\text{RegistryInit}}{\text{Registry}'}}{\forall p : Path \bullet \exists RepoInit \bullet repo' p = \theta Repo'}$$

We define a promotion schema which operates on a single repository in a registry.

$$\frac{\frac{\text{RegistryPromote}}{\Delta Registry \quad \Delta Repo \quad p? : Path}}{p? \in \text{dom } repo \quad \theta Repo = repo p? \quad repo' = repo \oplus \{p? \mapsto \theta Repo'\}}$$

We then promote the push and pull operations.

$$\begin{aligned} RegistryPush &\triangleq \exists \Delta Repo \bullet RepoPush \wedge RegistryPromote \\ RegistryPullOk &\triangleq \exists \Delta Repo \bullet RepoPullOk \wedge RegistryPromote \end{aligned}$$

Registries are arranged in a network indexed by hostname.

$$\frac{\text{Net}}{reg : Hostname \rightarrow Registry}$$

Initially, there are no registries in the network.

$$\frac{\text{NetInit} \quad \text{Net}'}{\text{reg}' = \emptyset}$$

We can add an empty registry to the network.

$$\frac{\text{NetAddRegistryOk} \quad \Delta \text{Net} \quad h? : \text{Hostname}}{h? \notin \text{dom reg} \quad \exists \text{RegistryInit} \bullet \text{reg}' = \text{reg} \cup \{h? \mapsto \theta \text{Registry}'\}}$$

We can also remove a registry from the network.

$$\frac{\text{NetRemoveRegistryOk} \quad \Delta \text{Net} \quad h? : \text{Hostname}}{h? \in \text{dom reg} \quad \text{reg}' = \{h?\} \triangleleft \text{reg}}$$

We define a promotion schema which operates on a single registry in a network.

$$\frac{\text{NetPromote} \quad \Delta \text{Net} \quad \Delta \text{Registry} \quad h? : \text{Hostname}}{h? \in \text{dom reg} \quad \theta \text{Registry} = \text{reg } h? \quad \text{reg}' = \text{reg} \oplus \{h? \mapsto \theta \text{Registry}'\}}$$

Finally, we promote the push and pull operations to work on a network.

$$\begin{aligned} \text{NetPushOk} &\triangleq \exists \Delta \text{Registry} \bullet \text{RegistryPush} \wedge \text{NetPromote} \\ \text{NetPullOk} &\triangleq \exists \Delta \text{Registry} \bullet \text{RegistryPullOk} \wedge \text{NetPromote} \end{aligned}$$

Pushing can fail if there is no registry with the input hostname.

7 Image References

An image reference identifies an image in a registry.

Ref

host : *Hostname*

path : *Path*

tag : *Tag*

dig : *OptionalContentDigest*

A tag is always logically present, but if it is omitted from the textual representation of an image reference, it defaults to *Latest*. A content digest may be part of an image reference or may be omitted.

So far the push and pull operations have accumulated several input parameters.

PushParms

h? : *Hostname*

p? : *Path*

t? : *Tag*

d? : *ContentDigest*

PullParms

h? : *Hostname*

p? : *Path*

t? : *Tag*

d? : *OptionalContentDigest*

An image reference is mapped to push input parameters as follows.

RefPushParms

r? : *Ref*

PushParms

h? = *r?.host*

p? = *r?.path*

t? = *r?.tag*

r?.dig = *None*

Pushing is not allowed with a content digest in the image reference.

An image reference is mapped to pull input parameters as follows.

$RefPullParms$	_____
$r? : Ref$	
$PullParms$	
$h? = r?.host$	
$p? = r?.path$	
$t? = r?.tag$	
$d? = r?.dig$	

Push can then be reframed to take an image reference.

$$RefPushOk \triangleq \exists PushParms \bullet NetPushOk \wedge RefPushParms$$

$$RefPullOk \triangleq \exists PullParms \bullet NetPullOk \wedge RefPullParms$$

A Z Notation

Numbers:

\mathbb{N} Natural numbers $\{0, 1, \dots\}$

Propositional logic and the schema calculus:

$\dots \wedge \dots$	And	$\langle\langle \dots \rangle\rangle$	Free type injection
$\dots \vee \dots$	Or	$[\dots]$	Given sets
$\dots \Rightarrow \dots$	Implies	$', ?, !, 0 \dots 9$	Schema decorations
$\forall \dots \mid \dots \bullet \dots$	For all	$\dots \vdash \dots$	theorem
$\exists \dots \mid \dots \bullet \dots$	There exists	$\theta \dots$	Binding formation
$\dots \setminus \dots$	Hiding	$\lambda \dots$	Function definition
$\dots \hat{=} \dots$	Schema definition	$\mu \dots$	Mu-expression
$\dots == \dots$	Abbreviation	$\Delta \dots$	State change
$\dots ::= \dots \mid \dots$	Free type definition	$\Xi \dots$	Invariant state change

Sets and sequences:

$\{\dots\}$	Set	$\dots \setminus \dots$	Set difference
$\{\dots \mid \dots \bullet \dots\}$	Set comprehension	$\bigcup \dots$	Distributed union
$\mathbb{P} \dots$	Set of subsets of	$\# \dots$	Cardinality
\emptyset	Empty set	$\dots \subseteq \dots$	Subset
$\dots \times \dots$	Cartesian product	$\dots \subset \dots$	Proper subset
$\dots \in \dots$	Set membership	$\dots \text{ partition } \dots$	Set partition
$\dots \notin \dots$	Set non-membership	seq	Sequences
$\dots \cup \dots$	Union	$\langle \dots \rangle$	Sequence
$\dots \cap \dots$	Intersection	disjoint \dots	Disjoint sequence of sets

Functions and relations:

$\dots \leftrightarrow \dots$	Relation	\dots^*	Reflexive-transitive closure
$\dots \rightarrow \dots$	Partial function	$\dots (\dots)$	Relational image
$\dots \rightarrow \dots$	Total function	$\dots \oplus \dots$	Functional overriding
$\dots \mapsto \dots$	Partial injection	$\dots \triangleleft \dots$	Domain restriction
$\dots \mapsto \dots$	Injection	$\dots \triangleright \dots$	Range restriction
dom \dots	Domain	$\dots \trianglelefteq \dots$	Domain subtraction
ran \dots	Range	$\dots \trianglerighteq \dots$	Range subtraction
$\dots \mapsto \dots$	maplet		
$\dots \sim \dots$	Relational inverse		

Axiomatic descriptions:

<i>Declarations</i>
<i>Predicates</i>

Schema definitions:

<i>SchemaName</i>
<i>Declaration</i>
<i>Predicates</i>