Introduction to OCI Image Technologies

Serving Container

Keyang Xie
xiekeyang@huawei.com

Jitang Lei
leijitang@huawei.com

HUAWEI TECHNOLOGIES CO., LTD.

Presenters

Keyang Xie	Software engineer from Huawei Euler OS Dep maintains of OCI image tools
Jitang Lei	Software Engineer from Huawei Euler OS Dep, Core Maintainer of docker engine, Maintainer of Image-tool

Agenda

- **Introduction OCI image spec**

  - Introducton and Overview

  - Image layout

  - Image Manifest & Index

  - Image configurations and Filesystem Layers

- **Introduction OCI image tool**

  - Application of OCI tools

  - Features

  - Work Following

  - Concept

Introduction of Image Spec

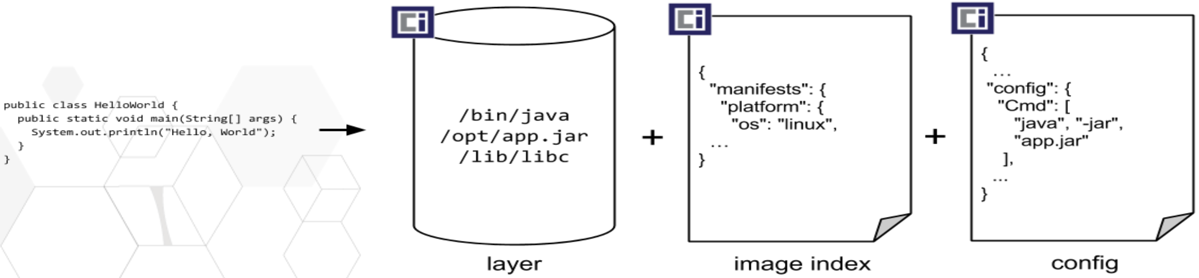
Introduction and overview

Goal of image spec

- Creates and maintains the software shipping container image format
- Enable the creation of interoperable tools for building, transporting, and preparing a container image to run.

What is an oci image

- an ordered collection of root filesystem changes and the corresponding execution parameters
- OCI Image consisting of a manifest, an optional image index, a set of filesystem layers, and a configuration.



```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World");
  }
}
```

```
/bin/java
/opt/app.jar
/lib/libc
```

```
{
  "manifests": {
    "platform": {
      "os": "linux",
  ...
}
```

```
{
  ...
 "config": {
  "Cmd": [
     "java", "-jar",
     "app.jar"
   ],
  ...
}
```

Media type

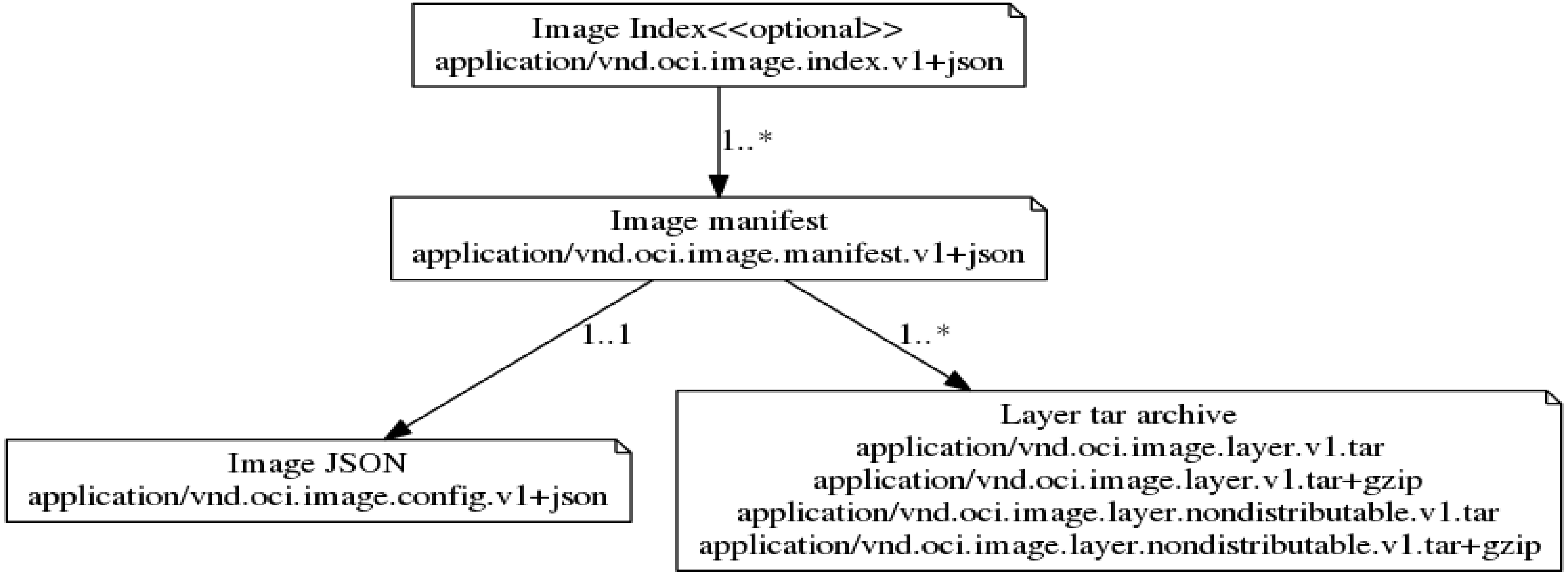


Image Index<<optional>>
application/vnd.oci.image.index.v1+json

1..*

Image manifest
application/vnd.oci.image.manifest.v1+json

1..1

1..*

Image JSON
application/vnd.oci.image.config.v1+json

Layer tar archive
application/vnd.oci.image.layer.v1.tar
application/vnd.oci.image.layer.v1.tar+gzip
application/vnd.oci.image.layer.nondistributable.v1.tar
application/vnd.oci.image.layer.nondistributable.v1.tar+gzip

Image Layout

- Content-addressable blobs and location-addressable references (refs).
- Given an image layout and a ref, a tool can create an OCI Runtime Specification bundle

The image layout is as follows:

- blobs directory
- oci-layout file
- index.json file

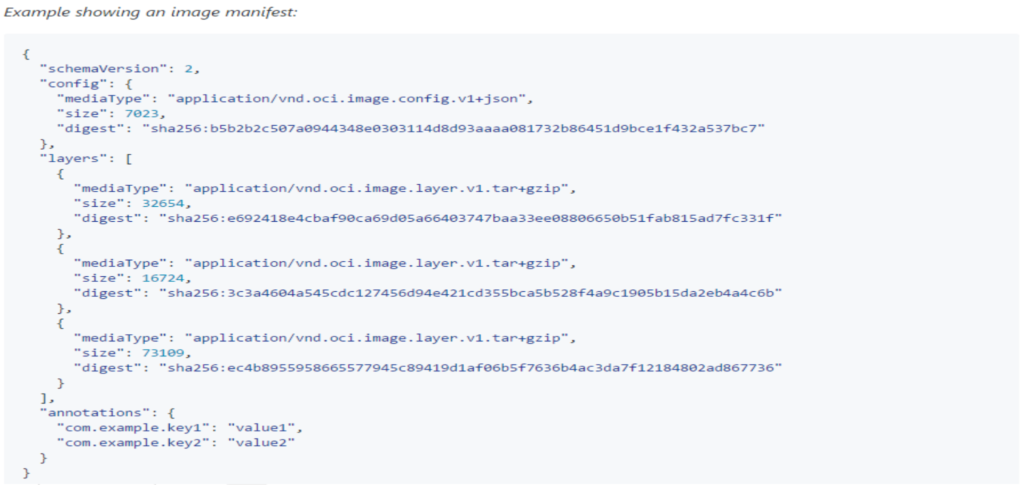
Layout Example

```
$ cd example.com/app/
$ find . -type f
./index.json
./oci-layout
./blobs/sha256/3588d02542238316759cbf24502f4344ffcc8a60c803870022f335d1390c13b4
./blobs/sha256/4b0bc1c4050b03c95ef2a8e36e25feac42fd31283e8c30b3ee5df6b043155d3c
./blobs/sha256/7968321274dc6b6171697c33df7815310468e694ac5be0ec03ff053bb135e768
```

Image manifest

- **Content-addressable images**
- **Multi-architecture images**
- **Be translatable to the OCI Runtime Specification**

An image manifest provides a configuration and set of layers for a single container image for a specific architecture and operating system.

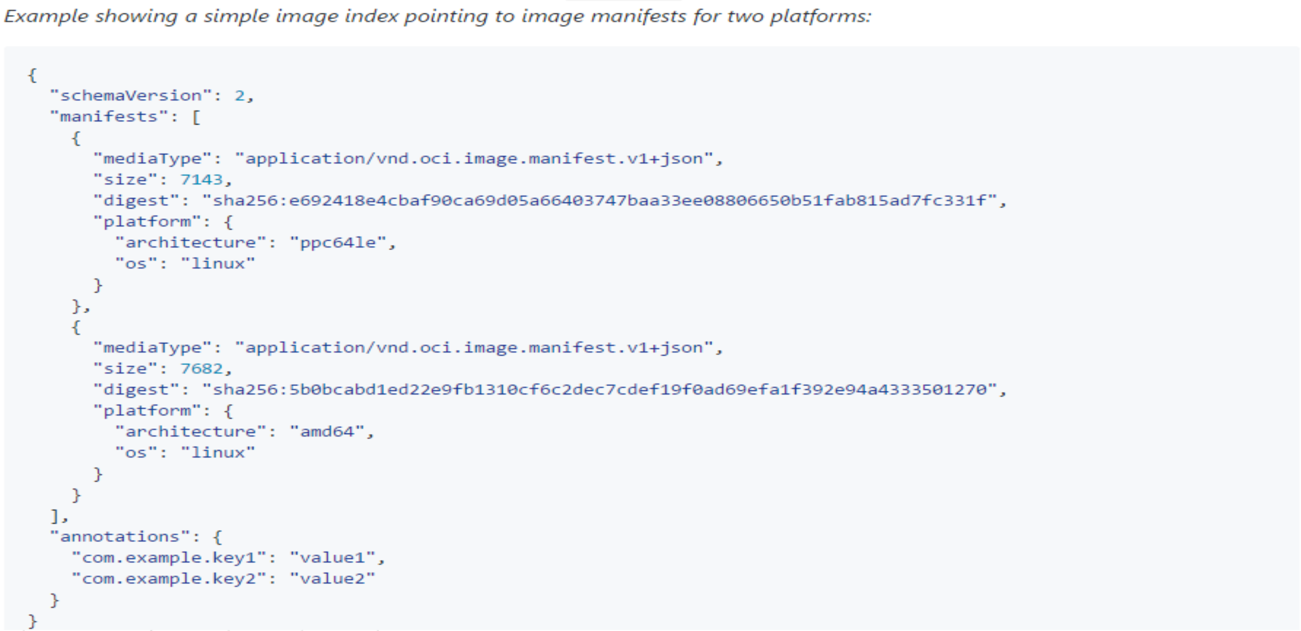


Example showing an image manifest:

```
{
  "schemaVersion": 2,
  "config": {
    "mediaType": "application/vnd.oci.image.config.v1+json",
    "size": 7023,
    "digest": "sha256:b5b2b2c507a0944348e0303114d8d93aaaa081732b86451d9bce1f432a537bc7"
  },
  "layers": [
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "size": 32654,
      "digest": "sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad7fc331f"
    },
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "size": 16724,
      "digest": "sha256:3c3a4604a545cdc127456d94e421cd355bca5b528f4a9c1905b15da2eb4a4c6b"
    },
    {
      "mediaType": "application/vnd.oci.image.layer.v1.tar+gzip",
      "size": 73109,
      "digest": "sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802ad867736"
    }
  ],
  "annotations": {
    "com.example.key1": "value1",
    "com.example.key2": "value2"
  }
}
```

Image index

The image index is a higher-level manifest which points to specific image manifests, ideal for one or more platforms.



Example showing a simple image index pointing to image manifests for two platforms:

```
{
  "schemaVersion": 2,
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "size": 7143,
      "digest": "sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad7fc331f",
      "platform": {
        "architecture": "ppc64le",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "size": 7682,
      "digest": "sha256:5b0bcabd1ed22e9fb1310cf6c2dec7cdef19f0ad69efa1f392e94a4333501270",
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    }
  ],
  "annotations": {
    "com.example.key1": "value1",
    "com.example.key2": "value2"
  }
}
```

Layers

- Set of filesystem changes.
- An archive of the files which have been added, changed, or deleted relative to its parent layer.
- One or more layers are applied on top of each other to create a complete filesystem.

How to create a layer

Change Types

- Additions
- Modifications
- Removals

Additions and Modifications are represented the same in the changeset tar archive.
Removals are represented using "whiteout" file entries

File Types

- regular files
- directories
- sockets
- symbolic links
- block devices
- character devices
- FIFOs

File Attributes

- Modification Time (mtime)
- User ID (uid)
- Group ID (gid)
- Mode (mode)
- Extended Attributes (xattrs)
- Symlink reference (linkname + symbolic link type)
- Hardlink reference (linkname)

Layers

Applying Changesets

- Layers Changesets are applied, rather than simply extracted as tar archives
- In the absence of any whiteout files in a layer changeset, the archive is extracted like a regular tar archive
- For existing files, removing the file path and recreating it based on the contents and arrtibutes in the layer
- Applying a layer changeset requires special consideration for the whiteout files

Whiteouts

Empty file with `.wh.` signifies a path should be deleted

Opaque Whiteouts

A file with name `.wh..wh.opq` indicating all siblings dir are hidden

Image Configuration

- JSON structure which describes some basic information
- references a cryptographic hash of each layer used by the image
- Immutable
- Changing it means creating a new derived image

ImageID

Each image's ID is given by the SHA256 hash of its configuration JSON.

DiffID

A layer DiffID is the digest over the layer's uncompressed tar archive

ChainID

ChainID identifies the subsequent application of those changesets.

Image Configuration

Properties

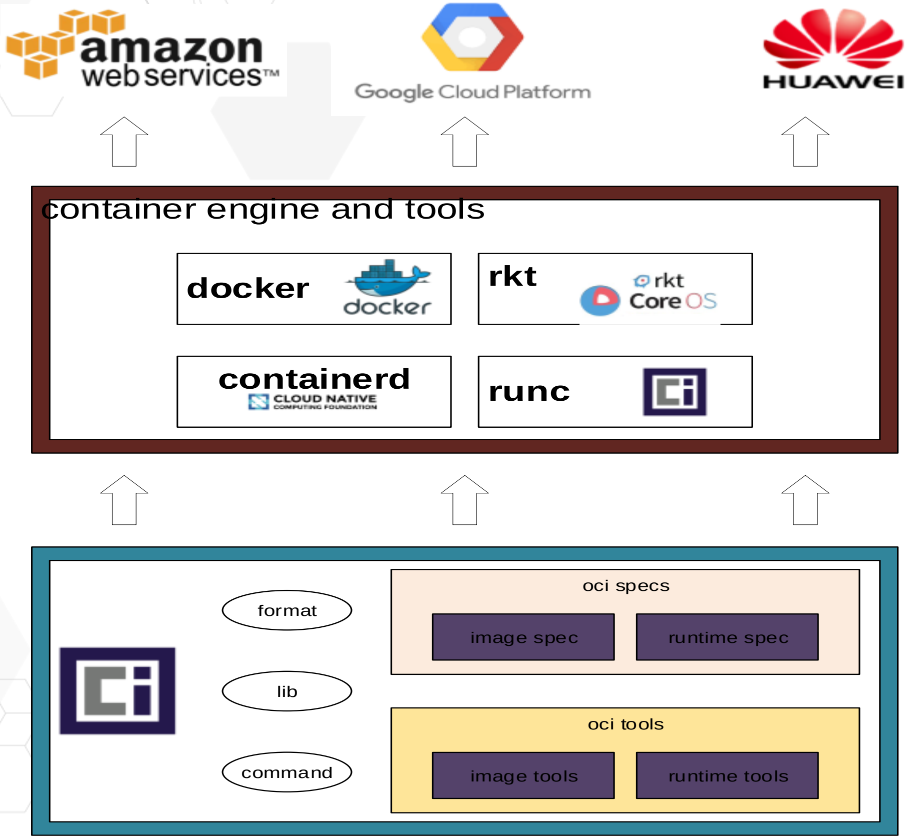
- Created
- Author
- Architecture
- Os
- Config
  - User
  - ExposedPorts
  - Env
  - Entrypoint
  - Cmd
  - Volumes
  - WorkingDir
  - Labels
  - StopSignal
- Rootfs
  - Type
  - Diff_ids
- History

Introduction of Image Tools

Application of OCI tools

- •Be collection of tools for working with the OCI image format specification.
- •Provides CLI commands, API functions so on, to support OCI consumers to utilize the canonical OCI image.
- •Provides immutable elements as serialization of file system of containers.
- •Release to V0.1.

Application of OCI tools



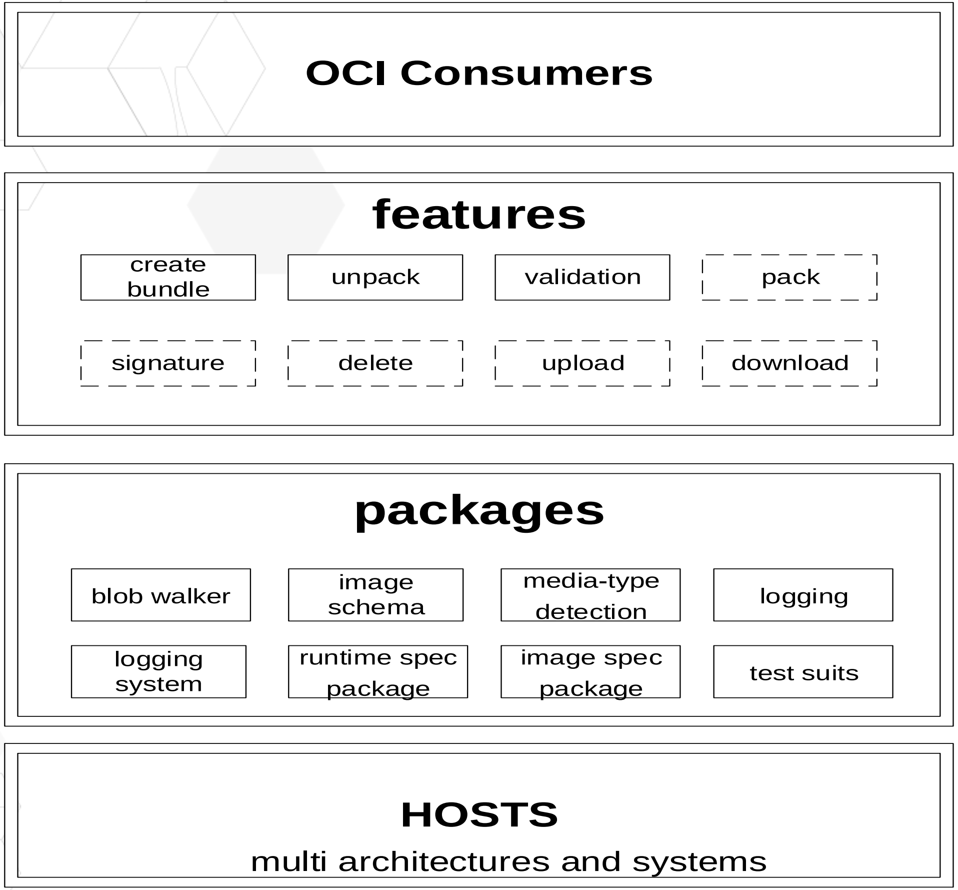
Features

Image tools help to resolve how bundle is expressed in the file system before running, and standardize the bundle on different disk file systems.

Feature	Description	status
Create	unpacks its layered filesystem, and translates the referenced config to a runtime-spec-compatible.	ready
Unpack	unpacks its layered filesystem.	ready
Validate	validates the given OCI file(s) against the OCI image specification.	ready
Pack	packs given layered filesystem, to generate OCI image bundle	plan
Upload	uploads OCI image to given web url via populer protocol like HTTP, CDN, so on.	plan
download	downloads OCI image from given web url.	plan
Signature	signature the OCI image to demonstrating the authenticity.	plan
Delete	delete given image from OCI layout.	plan

It provides hashing for content integrity. This is a generic requirement across almost all use cases to ensure content integrity, during creating, unpacking, validation, and so on.

Features



OCI Consumers

features

create bundle

unpack

validation

pack

signature

delete

upload

download

packages

blob walker

image schema

media-type detection

logging

logging system

runtime spec package

image spec package

test suits

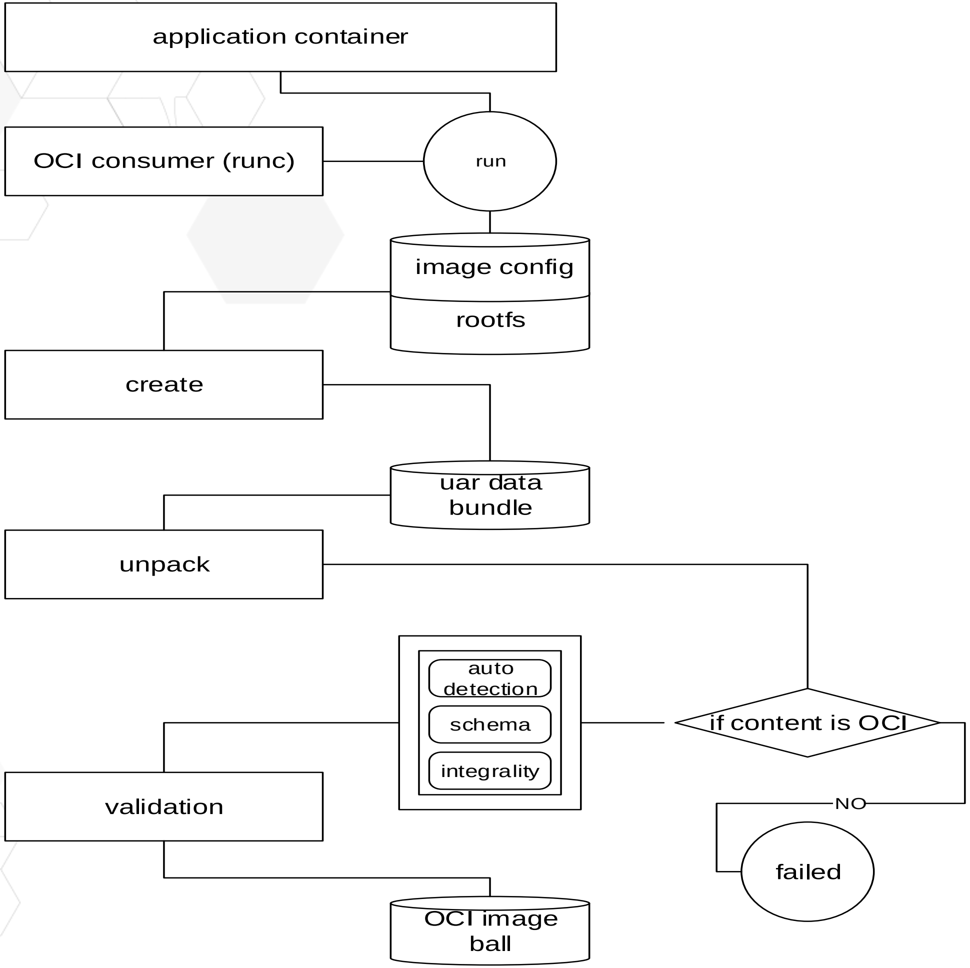
HOSTS

multi architectures and systems

Work flowing

- This section presents Work flowing of creating OCI runtime bundle.
- To validate OCI image data, and unpack it to user data if validation successful.
- Then create RootFS for it, and translating the configuration file from OCI image format to what runtime spec compatible.

Work flowing



Work flowing

Example:

```
$ skopeo copy docker://busybox oci:busybox-oci
$ mkdir busybox-bundle
$ oci-image-tool create --ref latest busybox-oci busybox-bundle
$ cd busybox-bundle && sudo runc run busybox
[...]
```

Concept

Image tools is a new repository in OCI, In future image tools should be more available

and present necessary functionalities to

OCI image would add more features. Here I present some brief concepts.

Concept

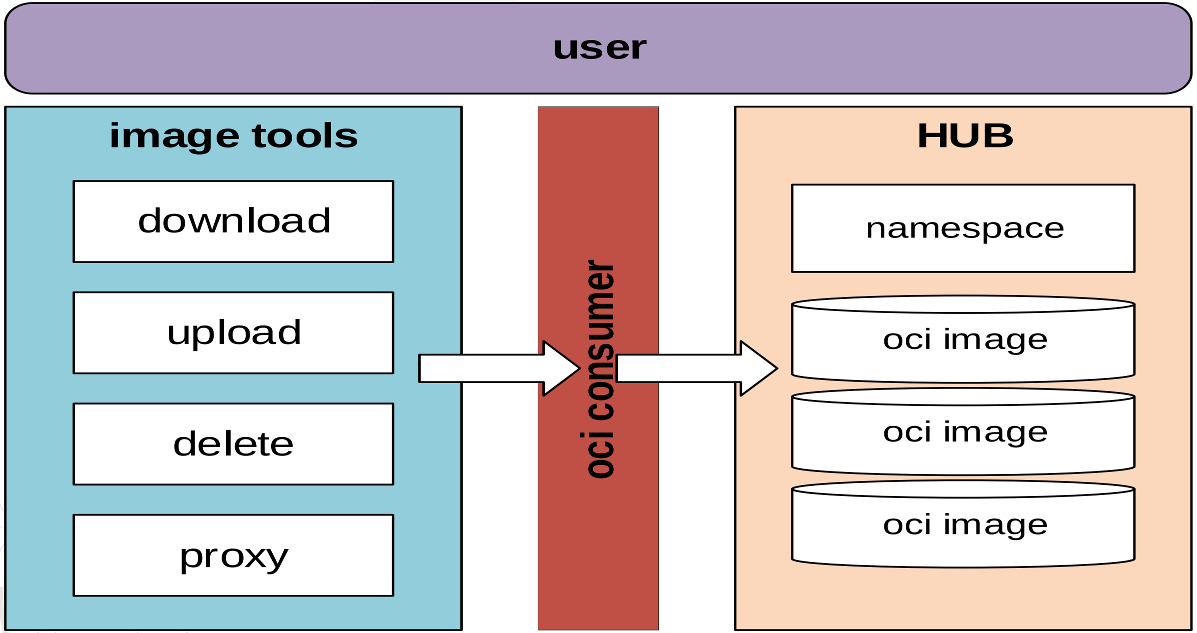
distribution

Different approaches to the intersection of naming and distribution make sense for different environments and are inherently controversial.

OCI should support multiple different naming & distribution schemes, including DNS-based, current Docker distribution/naming, IPFS, etc.

Concept

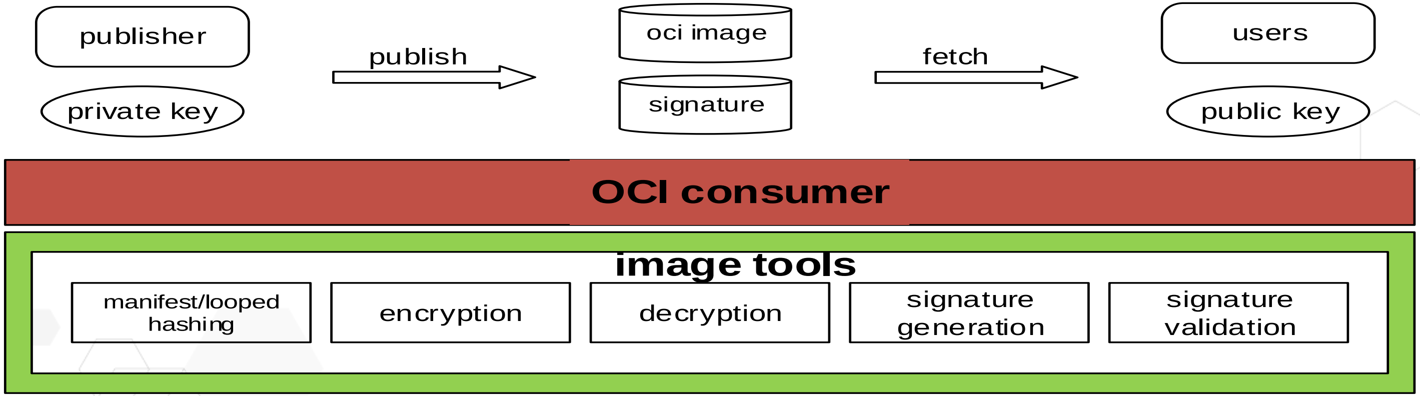
distribution



Concept

Signature

To make sure Image Content Trust. Provide a standardized cryptographic method for ensuring container content has not been altered. This may refer to GPG and TUF.



Reference Web Site

OCI web site: https://www.opencontainers.org

image spec repository: https://github.com/opencontainers/image-spec

image tools repository: https://github.com/opencontainers/image-tools

OCI Scope Table: https://www.opencontainers.org/about/oci-scope-table

Thank you!