Image Manifest V 2, Schema 2

Estimated reading time: 8 minutes

This document outlines the format of of the V2 image manifest, schema version 2. The original (and provisional) image manifest for V2 (schema 1), was introduced in the Docker daemon in the v1.3.0 release and is specified in the schema 1 manifest definition

This second schema version has two primary goals. The first is to allow multi-architecture images, through a "fat manifest" which references image manifests for platform-specific versions of an image. The second is to move the Docker engine towards content-addressable images, by supporting an image model where the image's configuration can be hashed to generate an ID for the image.

Media Types

The following media types are used by the manifest formats described here, and the resources they reference:

- `application/vnd.docker.distribution.manifest.v1+json`: schema1 (existing manifest format)
- `application/vnd.docker.distribution.manifest.v2+json`: New image manifest format (schemaVersion = 2)
- `application/vnd.docker.distribution.manifest.list.v2+json`: Manifest list, aka "fat manifest"
- `application/vnd.docker.image.rootfs.diff.tar.gzip`: "Layer", as a gzipped tar
- `application/vnd.docker.container.image.v1+json`: Container config JSON

Manifest List

The manifest list is the “fat manifest” which points to specific image manifests for one or more platforms. Its use is optional, and relatively few images will use one of these manifests. A client will distinguish a manifest list from an image manifest based on the Content-Type returned in the HTTP response.

Manifest List Field Descriptions

- **`schemaVersion`** *int*

  This field specifies the image manifest schema version as an integer. This schema uses the version `2`.

- **`mediaType`** *string*

  The MIME type of the manifest list. This should be set to `application/vnd.docker.distribution.manifest.list.v2+json`.

- **`manifests`** *array*

  The manifests field contains a list of manifests for specific platforms.

  Fields of an object in the manifests list are:

  - **`mediaType`** *string*

    The MIME type of the referenced object. This will generally be `application/vnd.docker.image.manifest.v2+json`, but it could also be `application/vnd.docker.image.manifest.v1+json` if the manifest list references a legacy schema-1 manifest.

  - **`size`** *int*

    The size in bytes of the object. This field exists so that a client will have an expected size for the content before validating. If the length of the retrieved content does not match the specified

length, the content should not be trusted.

- **`digest`** *string*

  The digest of the content, as defined by the Registry V2 HTTP API Specificiation.

- **`platform`** *object*

  The platform object describes the platform which the image in the manifest runs on. A full list of valid operating system and architecture values are listed in the Go language documentation for `$GOOS` and `$GOARCH`

  - **`architecture`** *string*

    The architecture field specifies the CPU architecture, for example `amd64` or `ppc64le`.

  - **`os`** *string*

    The os field specifies the operating system, for example `linux` or `windows`.

  - **`os.version`** *string*

    The optional os.version field specifies the operating system version, for example `10.0.10586`.

  - **`os.features`** *array*

    The optional os.features field specifies an array of strings, each listing a required OS feature (for example on Windows `win32k`).

  - **`variant`** *string*

The optional variant field specifies a variant of the CPU, for example `armv6l` to specify a particular CPU variant of the ARM CPU.

- **`features`** *array*

  The optional features field specifies an array of strings, each listing a required CPU feature (for example `sse4` or `aes`).

Example Manifest List

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

```
json { "schemaVersion": 2, "mediaType":
"application/vnd.docker.distribution.manifest.list.v2+json",
"manifests": [ { "mediaType":
"application/vnd.docker.image.manifest.v2+json", "size": 7143,
"digest":
"sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad
7fc331f", "platform": { "architecture": "ppc64le", "os": "linux",
} }, { "mediaType":
"application/vnd.docker.image.manifest.v2+json", "size": 7682,
"digest":
"sha256:5b0bcabd1ed22e9fb1310cf6c2dec7cdef19f0ad69efa1f392e94a433
3501270", "platform": { "architecture": "amd64", "os": "linux",
"features": [ "sse4" ] } } ] }
```

Image Manifest

The image manifest provides a configuration and a set of layers for a container image. It's the direct replacement for the schema-1 manifest.

Image Manifest Field Descriptions

- **`schemaVersion`** *int*

This field specifies the image manifest schema version as an integer. This schema uses version `2`.

- **`mediaType`** *string*

  The MIME type of the manifest. This should be set to `application/vnd.docker.distribution.manifest.v2+json`.

- **`config`** *object*

  The config field references a configuration object for a container, by digest. This configuration item is a JSON blob that the runtime uses to set up the container. This new schema uses a tweaked version of this configuration to allow image content-addressability on the daemon side.

  Fields of a config object are:

  - **`mediaType`** *string*

    The MIME type of the referenced object. This should generally be `application/vnd.docker.container.image.v1+json`.

  - **`size`** *int*

    The size in bytes of the object. This field exists so that a client will have an expected size for the content before validating. If the length of the retrieved content does not match the specified length, the content should not be trusted.

  - **`digest`** *string*

    The digest of the content, as defined by the Registry V2 HTTP API Specificiation.

- **`layers`** *array*

The layer list is ordered starting from the base image (opposite order of schema1).

Fields of an item in the layers list are:

- **`mediaType`** *string*

  The MIME type of the referenced object. This should generally be `application/vnd.docker.image.rootfs.diff.tar.gzip`.

- **`size`** *int*

  The size in bytes of the object. This field exists so that a client will have an expected size for the content before validating. If the length of the retrieved content does not match the specified length, the content should not be trusted.

- **`digest`** *string*

  The digest of the content, as defined by the Registry V2 HTTP API Specificiation.

- **`urls`** *array*

  For an ordinary layer, this is empty, and the layer contents can be retrieved directly from the registry. For a layer with *`mediatype`* of `application/vnd.docker.image.rootfs.foreign.diff.tar.gzip`, this contains a non-empty list of URLs from which this object can be downloaded.

Example Image Manifest

Example showing an image manifest: `json { "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "config":`

```
{ "mediaType": "application/vnd.docker.container.image.v1+json",
"size": 7023, "digest":
"sha256:b5b2b2c507a0944348e0303114d8d93aaaa081732b86451d9bce1f432
a537bc7" }, "layers": [ { "mediaType":
"application/vnd.docker.image.rootfs.diff.tar.gzip", "size":
32654, "digest":
"sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad
7fc331f" }, { "mediaType":
"application/vnd.docker.image.rootfs.diff.tar.gzip", "size":
16724, "digest":
"sha256:3c3a4604a545cdc127456d94e421cd355bca5b528f4a9c1905b15da2e
b4a4c6b" }, { "mediaType":
"application/vnd.docker.image.rootfs.diff.tar.gzip", "size":
73109, "digest":
"sha256:ec4b8955958665577945c89419d1af06b5f7636b4ac3da7f12184802a
d867736" } ], }
```

Backward compatibility

The registry will continue to accept uploads of manifests in both the old and new formats.

When pushing images, clients which support the new manifest format should first construct a manifest in the new format. If uploading this manifest fails, presumably because the registry only supports the old format, the client may fall back to uploading a manifest in the old format.

When pulling images, clients indicate support for this new version of the manifest format by sending the `application/vnd.docker.distribution.manifest.v2+json` and `application/vnd.docker.distribution.manifest.list.v2+json` media types in an `Accept` header when making a request to the `manifests` endpoint. Updated clients should check the `Content-Type` header to see

whether the manifest returned from the endpoint is in the old format, or is an image manifest or manifest list in the new format.

If the manifest being requested uses the new format, and the appropriate media type is not present in an `Accept` header, the registry will assume that the client cannot handle the manifest as-is, and rewrite it on the fly into the old format. If the object that would otherwise be returned is a manifest list, the registry will look up the appropriate manifest for the amd64 platform and linux OS, rewrite that manifest into the old format if necessary, and return the result to the client. If no suitable manifest is found in the manifest list, the registry will return a 404 error.

One of the challenges in rewriting manifests to the old format is that the old format involves an image configuration for each layer in the manifest, but the new format only provides one image configuration. To work around this, the registry will create synthetic image configurations for all layers except the top layer. These image configurations will not result in runnable images on their own, but only serve to fill in the parent chain in a compatible way. The IDs in these synthetic configurations will be derived from hashes of their respective blobs. The registry will create these configurations and their IDs using the same scheme as Docker 1.10 when it creates a legacy manifest to push to a registry which doesn't support the new format.