IBM

Open by design™

Beyond x86_64: Docker images for Multi-Platform

Phil Estes

IBM Cloud Open Technologies

Twitter: @estesp

About Me

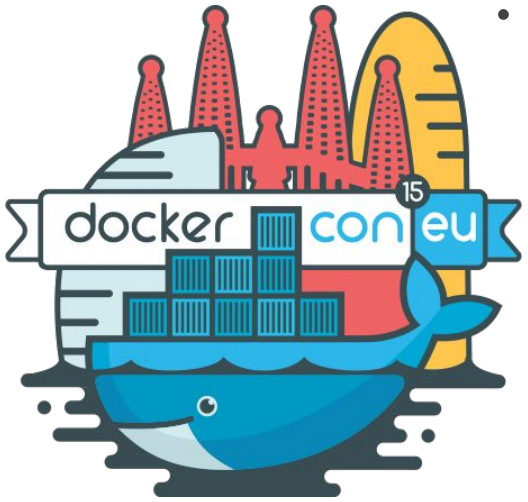
Phil Estes
Senior Technical Staff Member
IBM Cloud, Open Technologies
Container Strategy/Open Source Leader

Docker community core engine maintainer <
Linux/open source expertise for 15 years @ IBM <

Community activities & accomplishments

> Brought user namespace support to the Docker engine
> Helped design v2.2 image specification with multi-platform support
> Implemented first tool to create multi-platform images in Docker v2.3 registry
> Member of the “Docker Captains” program

Background: DockerCon Barcelona, Nov 2015



- **Multi-platform support in registry via "*workarounds*":**
  - Repo-per-arch/platform: **ppc64le/**, **s390x/**, **armv7/**
  - Image name prefixes: **rpi-** (rpi-node, rpi-consul, rpi-mysql)
  - Tag names with architecture detail:
    - multiarch/busybox**:s390x**
    - multiarch/busybox**:amd64**
    - multiarch/busybox**:armhf**

- **Nearly finalized v2.2 image manifest specification:**
  - PR #1068 defined the new specification format; proposed in Oct, agreed to and merged **Dec. 18th, 2015**
  - PR #1281 implemented the new spec format in the registry (docker/distribution), and was merged on **Jan. 7th, 2016**

The manifest list Registry Object Type

- How does the v2.2 image manifest spec support multi-platform?
  - Via a new registry manifest type: the **manifest list**
    - A manifest list contains **pointers** to existing manifest objects
    - A manifest list contains a **platform** specification associated with each pointer to an existing manifest
- What is a **platform** specification?

```
"platform": { // example with *all* fields
        "architecture": "amd64",
        "os": "linux",
        "os.version": "10.0.10586",
        "os.features": [
          "win32k"
        ],
        "variant": "armv6l"
        "features": [
          "sse4", "aes"
        ]
}
```

Could be very simple:

```
"platform": {
        "architecture": "ppc64le",
        "os": "linux",
}
```

Docker 1.10: Engine Support

- **Docker 1.10 included rudimentary engine support for manifest list objects:**
  - Engine checks manifest list entries against **`GOOS`** and **`GOARCH`** values from the Docker engine host platform
  - If a platform entry from the manifest list matches, that reference manifest is then pulled from the registry and used as that name:tag locally
- **Why rudimentary?**
  - Is not (yet) using the **`feature`** or **`variant`** fields to make determinations against the local host platform/architecture
  - Potentially more important for embedded platforms than server-oriented platforms

Creating a Manifest List

- **Registry v2.3 and above support manifest lists**
  - **DockerHub** now supports pushing manifest list objects
  - Open source "docker/distribution" v2.3 and above (this includes the official DockerHub "**registry**" image, now at v2.4.1)
- **Official tooling still under discussion**
  - Discussion: Where does it fit? (e.g. extend "**docker push**")
  - Details, details (how to specify inputs, image signing/trust, etc.)
- **Proof of concept tool available today for testing**
  - https://github.com/estesp/manifest-tool
  - Can inspect (list) or push manifest lists to private registry or DockerHub
  - Read more about it here: https://integratedcode.us/2016/04/22/a-step-towards-multi-platform-docker-images/

The Manifest Tool

- Input: YAML definition of input images + platform specifications

```
image: estesp/debian:jessie
manifests:
  -
    image: s390x/debian:jessie
    platform:
      architecture: s390x
      os: linux
  -
    image: debian:jessie
    platform:
      architecture: amd64
      os: linux
  -
    image: ppc64le/debian:jessie
    platform:
      architecture: ppc64le
      os: linux
  -
    image: aarch64/debian:jessie
    platform:
      architecture: arm64
      os: linux
      variant: armv8
```

- **image**: target manifest list **name:tag**
- **manifests**: list of image names with associated platform specification
  - **Architecture** and **OS** fields validated against known good values/combinations
- Image references are queried/validated; can reside in any repo, but on the same registry host
- Manifest list written to registry

Let's Demo

What's Left?

- **Engine: Full Support for 'platform' Definition**
  - Windows will exploit **os.version** and **os.features** entries for full platform matching
  - Linux-based engines need to determine how to "understand" **features** and **variant** with standardized definitions (e.g. `/proc/cpuinfo`)
- **Official tooling for manifest list push/inspection**
  - Requires full support for Docker Content Trust/image signing
  - Content publishers/creators need to be informed of capabilities
- **DockerHub/UI Updates**
  - Currently no visibility into "manifest list" entries
  - Platform information: a visual representation for consumers to find supported platforms for a given manifest list-based image
  - Workflow details (automated builds?)

IBM

Open by design™

Thank You!


@estesp


github.com/estesp


estesp@gmail.com


https://integratedcode.us


IRC: estesp