

Docker, container images and standards



Anthony Shaw

Follow

3 min read · Aug 1, 2016



3



What are Kelsey and [Solomon battling it out about?](#)

Docker and Open Source

Let me first off explain a couple of things, “Docker, Inc”, is a privately funded company, Solomon (and others) developed the Docker runtime, which runs and operates containerisation. If you’re not familiar with containerisation, stop reading and go research it and come back.

Up until now, when people have said they are “running containers”, it’s 99.9% likely they are talking about running Docker on a server. Docker is not the only player in town, see CoreOS rkt runtime, but they are by far and away the biggest.

You can also argue (but I won’t here) that Docker did not invent containerisation as a concept, but its recent popularity has been due to Docker’s improvements of the overall process of wrapping an application and its’ dependencies into an image.

<https://medium.com/@anthonypjshaw/docker-container-images-and-standards-ea92a60c087f>

The Docker image format is Open-Source (as in the code that reads and writes to the format), but it is not a standard. The image is the binary structure that keeps the application, files and dependencies that make a container.

Up to now, Docker Inc, has maintained the core docker engine as an open-source project, a significant amount of it's contributors are not Docker employees and it has been very open with the community about the roadmap for the core-engine.

Contrast this with other closed systems with proprietary formats from days gone by and you should be feeling quite comfortable.

I personally think Docker have done a wonderful job of keeping their core products open, involving the open-source community and not just doing "source-open" for kudos.

Now they also have a huge expectation from their private investors on commercialising the offering, adding value around the runtime and the packaging and services or support for the core IP. That will be a challenge for them, but they have some really smart people, so good luck to them.

What is Kelsey (Hightower) complaining about then?

Well, Docker is the runtime at the moment, but it will not always be that way. There is a huge-wave of tooling being developed at present around containerisation.

Fleet, Kubernetes, ContainerX, Rancher, CoreOS, all of these products are being built around this concept of containers. But because at the moment we only really have Docker, all of this is tightly coupled to the Docker image format. So if Docker were to change that for whatever reason, things would be very strange for a while *as if millions of voices suddenly cried out in terror, and were suddenly silenced*. If Docker were to be succeeded with another company with no standard, then we'll put ourselves back a year or so to reimplement everything all over again (Yey).

Also, one of the big advantages of containers is the ability to deploy applications seemingly consistently on any platform, which is awesome. But if another format comes along and that wonder becomes a chore, we will all wish we fixed this at the beginning.

Will the open-standard work?

That depends on the adoption of the primary vendors, the closest parallel I can draw is the OCCI standard, attempted to create a standard API for Clouds. Except that it wasn't adopted by any major provider (AWS, Microsoft, Open-Stack, VMware) so it's not a standard at all, its just a document talking about a world that doesn't exist.

Alternatively, there is the OVF standard, which is the file that describes a virtual machine, is an open-standard and widely adopted. However, the image format, VMDK, VHD (the important bit really) is a slightly different story. VMDK was made an open standard by VMware, but you have to convert VMDK images to a format to be run on other hypervisors because they were not designed to natively execute the format.

So there are a few options

1. Docker do it their way, everyone else does it theirs, Docker can choose not to support the open-standard or only support it via “conversion” tools
2. Docker works on an open, standard format and ditches their own in favour of the new way. New implementations of a container runtime use the standard format
3. Docker support their way and the standard way, (most likely) giving a tier-1 experience to their native format and supporting basic features in the standard format
4. Everyone slowly moves to another container runtime and forgets that any of this ever happened.

Time will tell.