Making compliance scalable in a container world

[draft of a presentation to the FSFE's Legal and Licensing Workshop, scheduled for April 15-17, 2020, but which has been cancelled]

David A. Levine
VP & Assistant General Counsel
Red Hat, Inc.

Scott K. Peterson
Senior Commercial Counsel
Red Hat, Inc.

Red Hat

- Have you heard about how software distributed in the form of container images presents compliance challenges?
- Let's bring container registries into the compliance mainstream:

  We have developed a mechanism for making corresponding source code available that exploits existing container registry capabilities.
- With the collaboration of others, we would like to refine this approach.

Container Technology - Motivation

- In enterprise computing, an application program is not a single executable file.
  - Most programs have a complex assortment of dependencies.
  - There can be a multitude of configuration details.
- Store and deploy as a unit: with container technology, a program is stored and deployed together with its dependencies.
- Speeds application development cycle by reducing conflicts between development and operations teams.
- Addresses the challenges of managing the deployment and operation of complex systems, even in a single data center.
- Hybrid Cloud: this value is even more compelling when managing deployment of workloads across multiple clouds.

Container Technology – Software Distribution

- Distribution of software in the form of container images is growing.
- Those who use containers to run their software find it useful to get software as pre-built container images, rather than doing all of the container assembly themselves.
- Distribution of container images differs significantly from distribution of software packages, such as RPMs.
- A package delivers a specific software component.
- A container image includes a large collection of software.

The unit of software distribution is changing
from one driven by
how the software is **built**
to one driven by
how the software is **used**.

Package Management Systems

- Package maintainers and package management tools have played a huge, under-appreciated role in source availability.
  - Maintainers build the software from source, and
  - That building is accompanied by packaging the corresponding source code.
- Result: Others don't even need to think about source availability.
  - The sources are available in the same unit as the delivery of the executable software and can be made available via the same distribution mechanism.
- The unit of software distribution (rpm or deb packages) is also the unit in which the software is **built**.
- Packages are building blocks: Packages represent components that are assembled into combinations in which the software is used.

Container images are different

- The software included in a container image is assembled based on how the software will be **used**, not on how software is built.
- A container image includes a diverse assortment of hundreds of software components.
  - While much of that content is assembled from packages, container images increasingly include non-package content.
  - Even for an image that features proprietary software, the image will typically include open source software, including GPL-licensed software.
- **There is no convention for making available the source code that corresponds to a container image.**

Container technology leads to different software distribution patterns

- With container technology, distribution is moving away from where the software is built (suppliers) toward where the software is used.
- It is increasingly common for container images to be made available in multiple registries.
- When a software supplier makes images available on other registries, how do they make the source code available?
- **Portable compliance matters** more than it has in the past.

Making Source Code Available

- Alternatives in GPL for commercial distribution
  - accompanied by the source code
  - written offer to provide the source code
  - equivalent access to copy – GPLv2, section 3, last paragraph
    - *If distribution of executable or object code is made by offering access to copy from a designated place, then **offering equivalent access to copy the source code from the same place** counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.*
- A registry-native approach to source availability – making source for container images available as container images:
  - facilitates providing **equivalent access** to the source.
  - facilitates **portable compliance**: making source available when container images are hosted in different registries.

Why offer source via a registry?

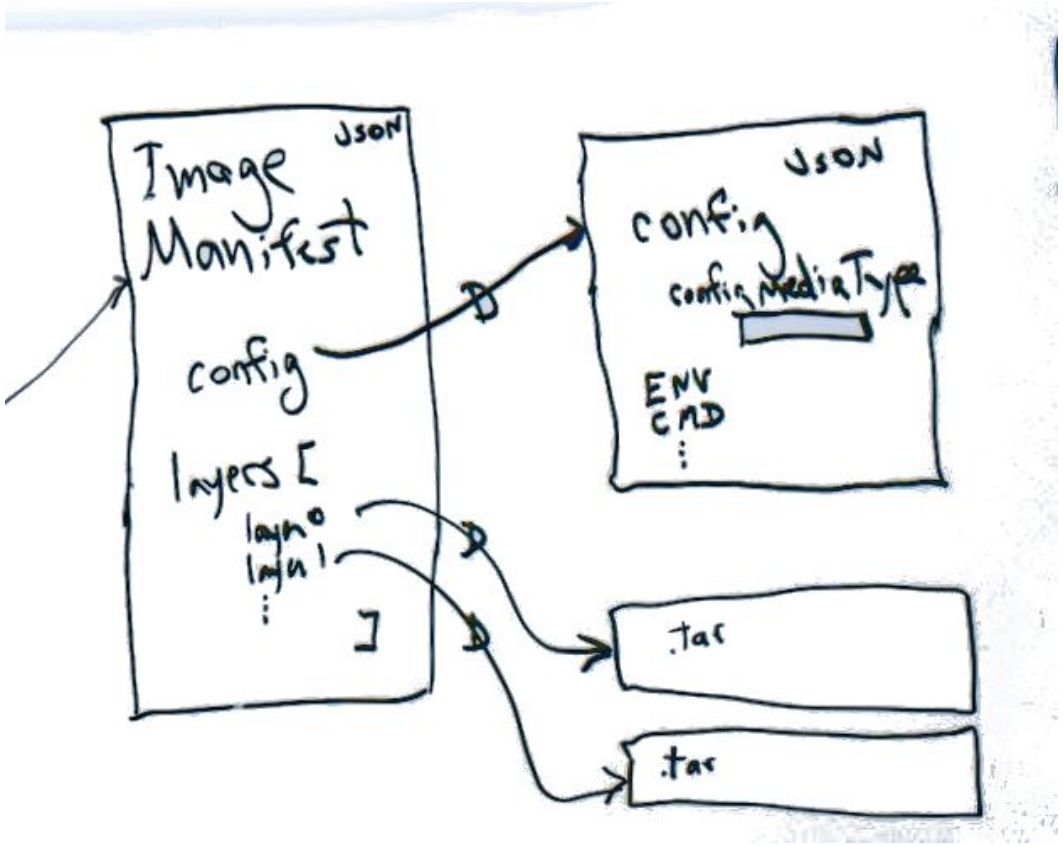
- We need **portable compliance**.
  - Container images typically include GPL-licensed.
  - A container image may be hosted in various registries.
  - We need a common technical mechanism that is available across registries, and that facilitates movement among registries.
- How might one do this? Imagine (the problems): 300 components in an image; a list that indicates where the source for each can be obtained. Perhaps all on one website. Where is that site? Who controls that site? How do the people running the various registries feel about distributing GPL-licensed software where the sources are at all sorts of other sites controlled by other people?
  - Portable?
  - How does one manage all that? Lots of work.
- GPL: "**equivalent access**"
  - How does that complex arrangement shape up when considering "equivalent access"?
  - Compare this level of complexity to making a source RPM available with an RPM.
- Compare this to making available via container image 'pulls' from the same registry.

As simple as getting source RPMs from the same package repository from which you get your executable RPMSs.

Source Code: Packages v. Container Images

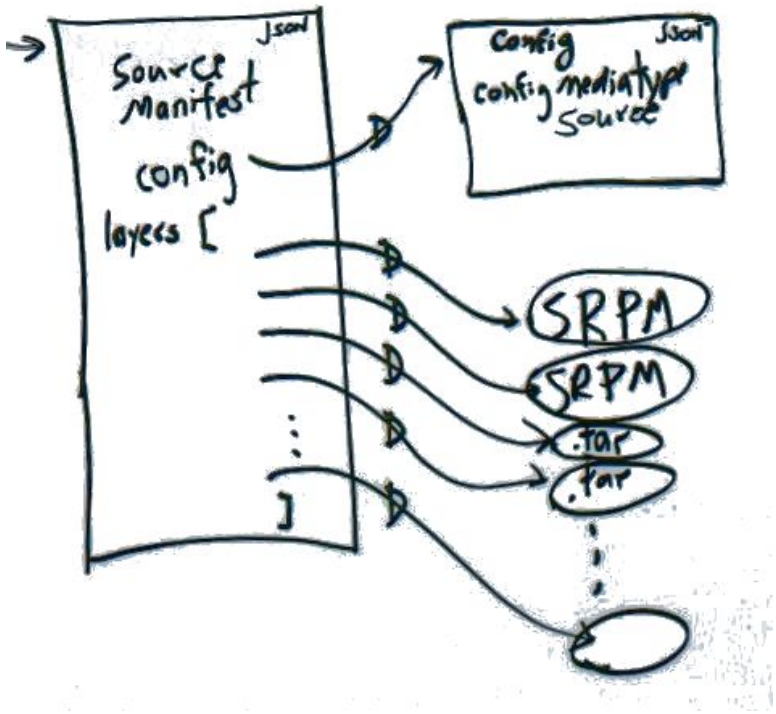
- Packages and container images might **seem similar**.
- How is the source code distribution challenge different for software that is distributed as container images (compared to software distributed as 'packages')?
  - Let's talk about Lego blocks: bricks v. castles.
  - Source code for components v. source for large collections of components
  - Analogy: packages = individual bricks; images = castle of hundreds of bricks.
  - A technical impediment: How much corresponding source do you need to store if:
    - stored per brick
    - stored per castle
  - The unit of packaging matters to a non-obvious degree.
- Many times more storage would be required when stored on a per-castle basis.
  - The next slides will show how, with the right level of **source artifact granularity** and attention to how the artifacts are represented, the **deduplication** inherent in the blob store can be used advantageously when hosting source code.

Images, Manifests & Registries



- Container **images** are distributed from servers that are known as container **registries**.
- Registries do not store "images" as single items.
- A container image is an assembly of parts.
- A registry stores and responds to requests for:
  - a list of the parts (the image **manifest**), and
  - each of the parts.
- A feature of container registries that is valuable when making source code artifacts available via a registry is that the parts of an image are stored in a content-addressable store, which by its nature provides automatic **deduplication**.

A registry-native approach to source code availability



- Source code is made available as a "container image" where the image manifest is a list of source artifacts (instead of a list of file system layers, as would be in a regular image). A "source container" is:
  - a list of the source artifacts
  - a source artifact for each of the software components that was used to build the image.
- A registry's **deduplication** capability can be used to great benefit in storing source code, **if** the unit of storage of source code **is <u>not</u> the source for an entire image**, but **<u>is</u> the source for each software component**.
  - Choice of **granularity** of source artifacts has huge impact on the amount of storage that is needed.

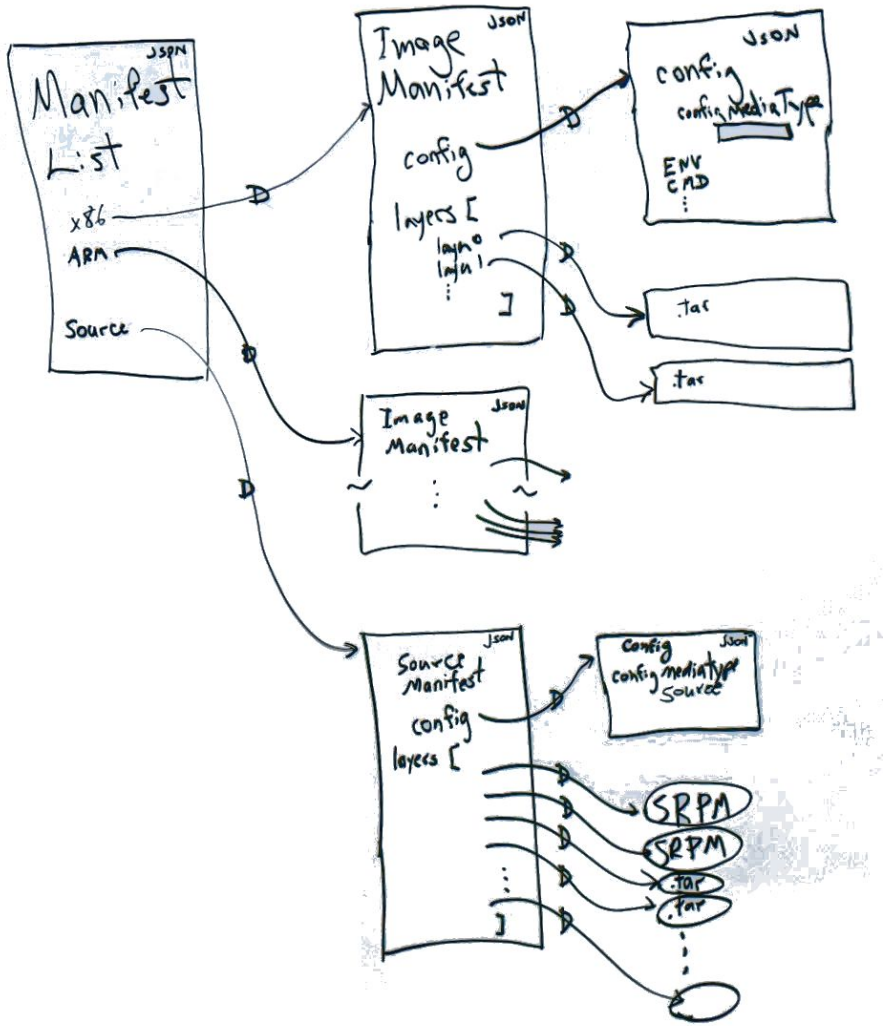
Initial Implementation

- An approach that can be supported by existing (source-unaware) registries
  - each source artifact is wrapped in a tarball and appears as a conventional container image 'layer'
- Tools have been developed to support this approach to source availability.
- Associating the source container with the corresponding executable image is using a labeling convention.

Improvements: Why make changes that will be visible to registries?

- **Successful deduplication** requires careful, consistent tar archives. To avoid that problem, registries need to be able to host the original source artifacts; we need more than just a regular layer mediatype.
- **Association of source with the corresponding executable image** via a labeling convention has limitations that can be overcome by listing the source image manifest in the image index.

Improvements: details



- Associate source and binary images
  - Including the source image in the list of architecture-specific images,
  - the source becomes a part of the overall container image.
- Mediatypes for source artifacts
- Work is ongoing at the Open Container Initiative

Where are we now?

- We have implemented the most portable solution.
- We still have work to do (collaboratively with others) to take this to the next level: [refer to work at OCI]
  - agree on container image format details to acknowledge source code content, so images don't need to masquerade as regular, executable container images.
  - image index: the container image format provides a way to directly refer to the corresponding source code.
- Is there anything else that we need to do for these images to be allowed into container registries everywhere?

Demo

Recap

Registry-native delivery of source provides

straightforward, portable

source availability compliance.

Call to Action

Vision: Portable compliance based on technical mechanisms that are available across registries.

We invite you to learn more and to help move toward this vision.

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat