Making compliance scalable in a container world: *container-registry-native support for compliance*

Scott K Peterson, Senior Commercial Counsel, Red Hat, Inc.
12 May 2021

FSFE Legal and Licensing Workshop - LLW 2021

Red Hat

Making compliance scalable in a container world: container-registry-native support for compliance

The growth of container technology leads to distribution of software via container registries, for which there is no established way to make source code available. To bring container registries into the compliance mainstream, we have developed a mechanism for making corresponding source code available through the same registry that delivers the executable container images; this mechanism exploits existing container registry capabilities. This builds compliance-related features into the technology, rather than expecting that compliance will be accomplished with external mechanisms: compliance by design. In addition, delivery of source along with the executable software can simplify other compliance requirements as well as provide security-related benefits.

Distribution of Software as Container Images

- A container image includes much more software than the featured application of the image (~100-400 other components).
  - Container technology is a complexity / storage tradeoff.
- That other software is very likely to include GPL-licensed software.
  - Does that lead to a copyleft problem for software in the image? No.
- But there is the GPL's source availability requirement.
- Should everyone who distributes their software via container images need to hire a team of compliance experts?
  - No. With "source containers" we seek to make compliance straightforward.

"Source Containers" — How it works

- Source containers is a technical mechanism for making source code available when software is distributed via container registries.
- When a regular executable container image is built, the corresponding source code is assembled arranged in a source container image.
- This source image can be moved along with the executable image and hosted in the same registries; source can travel with the executable image.
- Having built the source image at the beginning, subsequent source availability is straightforward.
- This builds compliance-related features into the technology, rather than expecting that compliance will be accomplished with external mechanisms.

"Container"

- The word "container" can be confusing:
  - The containment is a runtime experience, not a distribution experience.
  - It refers to using features built into the kernel to present a "contained" experience for code that is running.
- "Container images" are stored and distributed.
  - A container image contains an archive of a file system that is used to provision a container.
- A "source image" (also known as a "source container") is a container image in which the payload is source code, rather than executable code.
- A "container registry" is a service that is accessed by tools (it is **not** the GUI-oriented catalog that humans see).

Deeper into distribution of software as container images

- Tool-driven distribution
  - Human-readable websites (catalogs) are only indirectly related and are optional, not central to distribution.
- With distribution via container registries, an application may be more likely to be made available from multiple registries, not just from the supplier's single point of distribution.
- A container image is a manifest that has a list of references to blobs that contain the content:
  - configuration information and other metadata
  - "layers" that are serialized file systems that are laid down to become the filesystem used to provision a container.

Compliance by design

- Let's build compliance-related capabilities into container tooling.
- Container image distribution is all about tools.
  - Source availability should use those same tools, rather than depend on instructions for humans to follow
- Source code distribution should be managed natively along with the executable software.
- Compliance mechanisms should be able to travel with the software.
- There should be a compliance mechanism that is available on all registries.

Do container registries need source-code-specific features?

- **No**. Delivering source through container registries exploits existing container registry capabilities
- A **container image** is a manifest that references blobs.
- A registry is a service for delivering those manifests, then the blobs.
- A **source image** is a manifest, where the referenced blobs are source code artifacts (like source RPMs, or tarballs, or JAR files).
  - Granularity has huge impact on deduplication: one RPM per "layer"
- Source code can be made available along with the container image, and tools can access the registry to download the elements that a user wants.

Challenge #1 - manifests and blobs that registries don't reject

- Registries have policies that restrict content.

- Current
  - Masquerade as a regular image, such as by wrapping each source artifact in a tarball and using existing mediaTypes.
- Future
  - Conform to the OCI Artifacts project's way of representing non-standard image content.

Challenge #2: associating the source and executable images

- Given an executable container image, how does one identify the corresponding source image?

- Current
  - Naming convention (similar to what is done with SRPMs) adding "-source" to an image tag.
- Future?
  - We have explored other ways
    - For example, listing the source image in the image index (a manifest of manifests). But, there are implications for registry operation.
  - For the moment, continuing to use the naming convention.

Where are we?

- We been building source images in our software production pipeline for over a year.
- We have source images in our registry for over half of our product images.
  - for example, Red Hat Universal Base Images (UBI)
  - rollout to each of our products is ongoing
- Work is ongoing to replace the initial tool that we have been using for building source images.
  - That work will add source image capabilities to *buildah* (a tool for building OCI-compliant container images).
  - The new tool will enable us to adjust format details, such as to conform to expectations set in the OCI artifacts project.

Identification (source images can help)

- I have been describing a mechanism for **delivery** of source code.
- But what about **identification** of the corresponding source code?
- Red Hat builds its container images from the base on up. Thus, RH is in a position to know which components were used to build the image.
- But what about those who build on a base image built by someone else?
- Source images to the rescue:
  - The creator of the base image should also provide a source image.
  - The source image should be available in the same way that you obtained the base image.
  - You should be able to use that source image as the starting point for a source image that goes with your executable image.

Registry delivery of source code – there's more!

- If you have a mechanism that provides for delivery of the source code, why would you separate those notices from their context in the source code?
  - *An economically efficient model for open source software license compliance* by Jeff Kaufman https://opensource.com/article/17/9/economically-efficient-model
  - *The source code is the license* by Scott K Peterson https://opensource.com/article/17/12/source-code-license
  - *Is open source software licensing broken?* by Scott K Peterson https://opensource.com/article/20/2/open-source-licensing

- Want to understand what's in your container images?
  - What if you had the source code?

Why deliver the source code through a container registry?

- **Efficient compliance**. Creating a source availability artifact (a source image) when the executable image is built; manage that source image using the same tools as are used to manage the executable images.
- **Portable compliance**. The same tooling that moves the executable images can be used to move, store, and serve the source images - *on whatever registry an image is hosted*.
- **Equivalent access**. Making source for container images available as container images is certainly a good way of providing equivalent access to the source: get executable via a registry; get corresponding source via a registry.

GPL, source code availability, equivalent access

- GPL requires that, when software is distributed, the corresponding source code be made available.
- Such as:
  - **Accompany it with the source code**
  - Provide a written offer to provide the source code; or
  - **Provide equivalent access to copy***

    "If distribution . . . 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."

*GPLv2, section 3, last paragraph

Let's design source code availability into container tools and processes to facilitate compliance that is efficient and portable.

"Making compliance scalable in a container world"
https://opensource.com/article/20/7/compliance-containers
"What's in a container image: Meeting the legal challenges"
https://opensource.com/article/18/7/whats-container-image-meeting-legal-challenges

Questions?

"Making compliance scalable in a container world: container-registry-native support for compliance"
Scott K Peterson, Senior Commercial Counsel, Red Hat
LLW 2021 - 12 May 2021

CHR waived