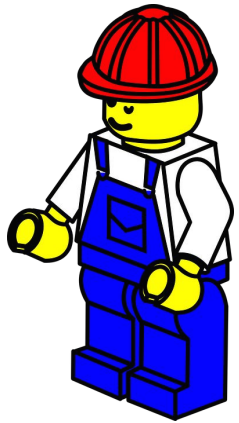


Introducing Buildah:
a tool for building OCI container images

OpenShift Commons Briefing

Nalin Dahyabhai @nalind
Dan Walsh @rhatdan

What even *is* an image?

- Described by a *manifest*, which enumerates:
  - Digest and size of the *configuration blob*, typically a JSON-encoded structure.
    - Configuration for running a container.
  - Digest and size of *layer blobs*, which are combined to produce the initial filesystem for a container.
- Not a single format, multiple closely-related and similar formats.

Configuration Blobs

- Always a JSON-encoded structure.
- Includes fields for, among other things:
  - Environment variables
  - Working directory
  - Command to run
  - User to run it as
- Not the same format as configuration that can be passed to *runc*.
- No cgroup, process limit, namespace settings, etc.

Layers

- Notionally, you can build up a container's root filesystem by extracting them all, one on top of another.
- Layers are basically tarballs.
  - *Whiteout* (a zero-length file with a name that matches a pattern) is used to indicate "delete a file or directory" whose name can be computed from that pattern.
- Can be reused among images — space savings!

How do you run a container?

- Set up a root filesystem using the layers.
- Use the configuration blob to build a configuration for your runtime, for example *runc* or a compatible tool.
- Start it up.

Building an image from a container, the long part

- Determine what changed in its filesystem, relative to its source image.
- Build a new layer containing those changes.

How do you know what changed in a container?

- If you used a union filesystem to set up the container's root, do something clever.
  - You can re-use the storage for the image's contents, which the container "inherits" — space savings!
- If you didn't use a union filesystem, walk the container's tree and a "clean" one based on the image.

Building an image from a container, all the parts

- Determine what changed in its filesystem, relative to its source image.
- Tar up the additions, with *whiteout* for deleted items, call it a new layer.
- Add the layer to the original image's manifest, if there was one, or generate a new manifest.
- Produce a new *configuration blob* using the container's settings and any that were inherited from an original image.

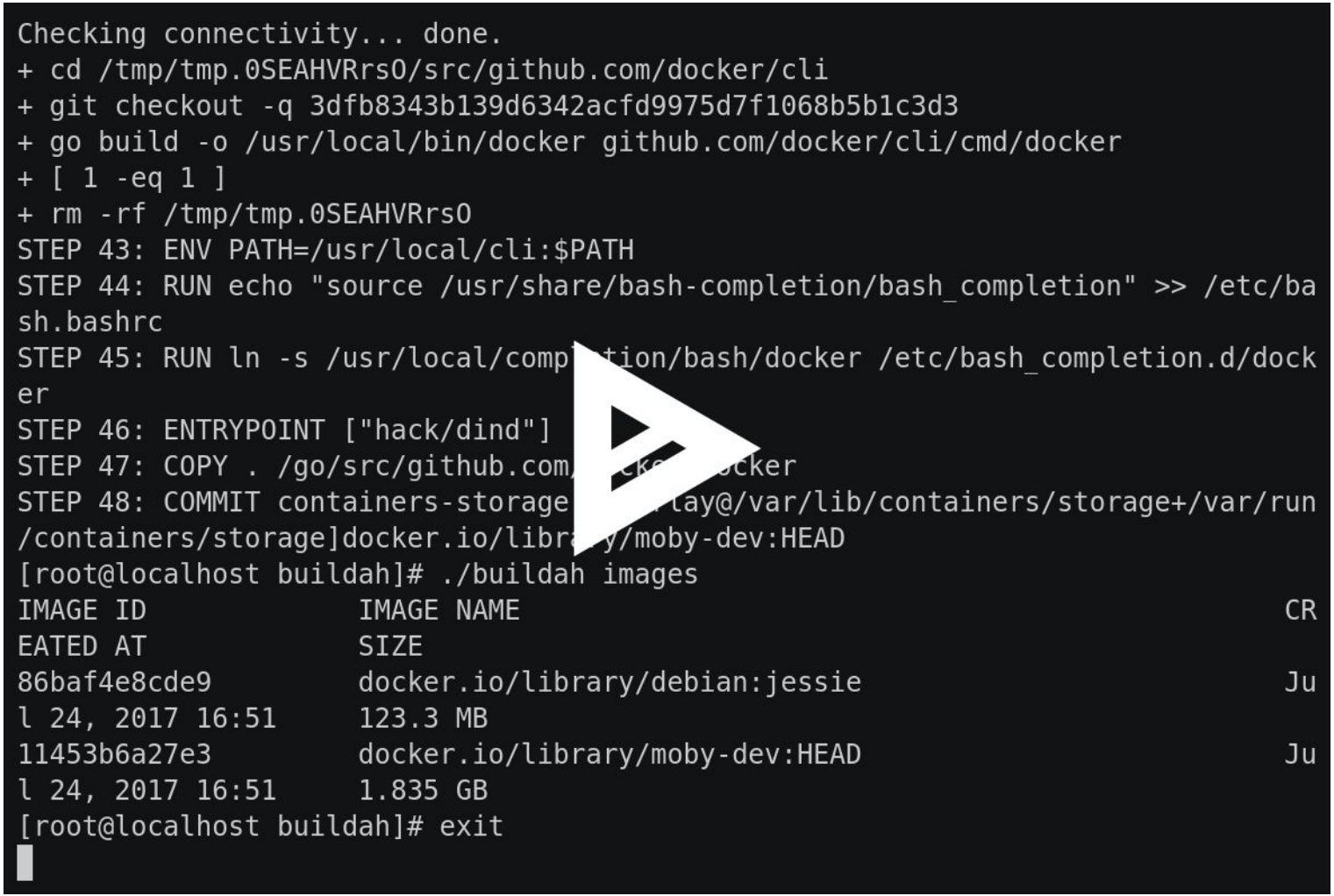
So what does buildah do?

- *buildah from*
  - Build up a container root filesystem from an image or *scratch*.
- *buildah config*
  - Adjust defaults in the image's configuration blob.
- *buildah run*
  - Run a command in the container's filesystem using *runc*.
- *buildah mount*
  - Mount the container's root filesystem on the host.
- *buildah commit*
  - Use the container's changes wrt its image to build a new image.

Other things buildah does

- Share libraries and on-disk storage with CRI-O
- Push images to registries or a local *dockerd* instance
  - *buildah push*
- Build images using a Dockerfile for instructions
  - *buildah build-using-dockerfile (a.k.a. buildah bud)*
- ~~Parallel park~~
- Oh, it also unmounts container filesystems
  - *buildah unmount*
- Provide a library API that’s used by the CLI
- ~~Do the heavy lifting in a daemon that the CLI connects to~~

```
licenses/doge
+ cp -pr LICENSE /root/rpmbuild/BUILDROOT/doge-3.5.0-4.fc26.x86_64/usr/share/lic
enses/doge
+ exit 0
Provides: doge = 3.5.0-4.fc26 python3.6dist(doge) = 3.5.0 python3dist(doge) = 3.
5.0
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDigests) <=
4.6.0-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1
Requires: /usr/bin/python3 python(abi) = 3.6
Checking for unpackaged file(s): /usr/lib/rpm/check-files /root/rpmbuild/BUILDRO
OT/doge-3.5.0-4.fc26.x86_64
Wrote: /root/rpmbuild/RPMS/noarch/doge-3.5.0-4.fc26.noarch.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.q9xfyy
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd doge-3.5.0
+ /usr/bin/rm -rf /root/rpmbuild/BUILDROOT/doge-3.5.0-4.fc26.x86_64
+ exit 0
Executing(--clean): /bin/sh -e /var/tmp/rpm-tmp.vjq416
+ umask 022
+ cd /root/rpmbuild/BUILD
+ rm -rf doge-3.5.0
+ exit 0
[root@localhost buildah]# cp -v $root/root/rpmbuild
```


Checking connectivity... done.
+ cd /tmp/tmp.0SEAHVRrsO/src/github.com/docker/cli
+ git checkout -q 3dfb8343b139d6342acfd9975d7f1068b5b1c3d3
+ go build -o /usr/local/bin/docker github.com/docker/cli/cmd/docker
+ [1 -eq 1]
+ rm -rf /tmp/tmp.0SEAHVRrsO
STEP 43: ENV PATH=/usr/local/cli:$PATH
STEP 44: RUN echo "source /usr/share/bash-completion/bash_completion" >> /etc/ba
sh.bashrc
STEP 45: RUN ln -s /usr/local/comp tion/bash/docker /etc/bash_completion.d/dock
er
STEP 46: ENTRYPOINT ["hack/dind"]
STEP 47: COPY . /go/src/github.com/ cke docker
STEP 48: COMMIT containers-storage lay@/var/lib/containers/storage+/var/run
/containers/storage]docker.io/libr y/moby-dev:HEAD
[root@localhost buildah]# ./buildah images
IMAGE ID IMAGE NAME CR
EATED AT SIZE
86baf4e8cde9 docker.io/library/debian:jessie Ju
l 24, 2017 16:51 123.3 MB
11453b6a27e3 docker.io/library/moby-dev:HEAD Ju
l 24, 2017 16:51 1.835 GB
[root@localhost buildah]# exit

More information

Source code: https://github.com/projectatomic/buildah

Blog posts: http://www.projectatomic.io/blog/, https://medium.com/cri-o

IRC: currently squatting in #cri-o on freenode

- nalind (@nalind)
- dwalsh (@rhatdan)