MULTI-ARRRRCH

Building images for multiple architectures using buildah

Nalin Dahyabhai
Red Hat
DevConf.CZ 2020

Red Hat

Here's What We're Covering Today

- Building and pushing single-arch images to registries
- Pulling single-arch images from registries
- What's in a manifest
- Pulling multi-arch images from registries
- What's in a manifest list
- Building and pushing multi-arch images to registries
- `buildah manifest`
- Using `buildah manifest`
- Building those images for those arches

Building and pushing single-arch images to registries

- Build an image.
- Push image to a repository in a registry, either using a tag, or noting its digest.
- Tell your friends about your image using the `registry/repository:tag` or `registry/repository@digest` name.
- Destroy the local evidence, never speak of this again.

Pulling single-arch images from registries

- Client asks registry for the manifest for an image that matches a given digest or tag in a repository.
- Registry hands back manifest for the image.
- Client asks for the layers and other things that make up the image.
- Client builds root filesystem, sets up namespaces, etc., execs the entry point or command.

What's in a manifest anyway?


```
{
      "schemaVersion": 2,
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "config": {
            "mediaType": "application/vnd.docker.container.image.v1+json",
            "size": 2199,
            "digest": "sha256:f0858ad3febdf45bb2e5501cb459affffacef081f79eaa436085c3b6d9bd46ca"
      },
      "layers": [
            {
                  "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
                  "size": 66774261,
                  "Digest": "sha256:d318c91bf2a81634e0283fb7e7362efdd7c21164b60b74498360756dc82a95d9"
            }
      ]
}
```


Red Hat

Pulling multi-arch images from registries

- Client asks registry for the manifest for an image that matches a given digest or tag in a repository.
- Registry hands back a *list* of manifests.
- Client selects an image from the list.
- Client asks registry for the manifest for the chosen image.
- Registry hands back manifest for the image.
- Client asks for the layers and other things that make up the image.
- Client builds root filesystem, sets up namespaces, execs the entry point or command.

What's in the manifest list anyway?

```
{
      "schemaVersion": 2,
      "manifests": [
            {
                  "mediaType": "application/vnd.oci.image.manifest.v1+json",
                  "digest": "sha256:2bdf5f7e128130d711351178b9daaeb5e7c1cc63b3738882fc2cad80245b21eb",
                  "size": 500,
                  "platform": {
                        "architecture": "amd64",
                        "os": "linux"
                  }
            }
      ]
}
```

Red Hat

Building and pushing multi-arch images to registries

- Build an image. Repeat, repeat, repeat.
- Push images to a repository in a registry, either using unique tags, or noting their digests.
- Build a list of those images, describing the types of systems they're each best for.
- Push the list to a registry, probably using a tag, or noting its digest.
- Tell your friends about your “image” using the `registry/repository:tag` or `registry/repository@digest` name.
- Destroy the local evidence, never speak of this again.

buildah manifest

- `buildah manifest create` *`list`*
  - Creates a new “image” that’s actually an image index / manifest list.
- `buildah manifest add` *`list image`*
  - Adds an entry to the list.
  - Handles either local (names or IDs) or remote (`docker://`…) images.
- `buildah manifest push` *`list registry/repository:tag`*
  - Pushes just the list.
- `buildah manifest push --all` *`list registry/repository:tag`*
  - Pushes the list *and* everything it references.
- `buildah rmi` *`list`*
  - Removes the list from local storage.

Using buildah manifest

- `buildah manifest add` takes arguments.
  - `--arch`: mostly Go arch names.
  - `--os`: *linux*
  - `--os-version`: mostly unused, except maybe when OS is ***`windows`***.
  - `--variant`: mostly unused, except for ARM.
  - `--features`: unused, in Docker format but reserved in OCI format.
  - `--os-features`: unused, except maybe when OS is ***`windows`***.
- Read from the image by default, which is usually enough.
  - It's not enough for ARM, and we need to do some work there.

Building those images for those arches

- Build on actual hardware, push to registry, for each arch. Build list, push list.
  - Add images using their digests or arch-specific tags.
  - The fastest or second-fastest option.
- Build in a VM, push to registry, for each arch. Build list, push list.
  - Cross-arch VMs exist, are kind of slow.
  - The slowest option.
- Cross-compile on build host for runtime arch, install onto a suitable base image, push along with list.
  - Be sure to mark the image with the right architecture!
  - The second-fastest or fastest option.
- Build for runtime, emulate for RUN (qemu-user-static, more like qemu-user-*magic*, amirite?), push along with list.
  - Faster than a cross-arch VM, probably slower than actual hardware.

fedora-with-iputils (1/2)

```
#!/bin/bash

cat > Dockerfile <<- EOF
    FROM fedora
    RUN dnf -y install iputils && dnf clean all
EOF

declare -A image
for arch in amd64 arm64 s390x ; do
    buildah bud --override-arch $arch --iidfile iid .
    image[$arch]=$(cat iid)
done
```

fedora-with-iputils (2/2)

```
buildah manifest create fedora-with-iputils

for arch in ${!image[@]} ; do
     flags=
     case $arch in
     arm64|aarch64)
          flags="--arch arm64 --variant v8"
     esac
     buildah manifest add fedora-with-iputils $flags ${image[$arch]}
done

buildah manifest push --all fedora-with-iputils \
docker://nalind/fedora-with-iputils
```

cursed-image

```
buildah manifest create cursed-image
buildah manifest add --override-arch=arm64   --override-os=linux \
--os=linux --arch=arm64 --variant=v8 cursed-image docker://fedora
buildah manifest add --override-arch=amd64   --override-os=linux \
--os=linux --arch=amd64              cursed-image docker://centos
buildah manifest add --override-arch=ppc64le --override-os=linux \
--os=linux --arch=ppc64le             cursed-image docker://debian
buildah manifest add --override-arch=s390x   --override-os=linux \
--os=linux --arch=s390x              cursed-image docker://alpine
buildah manifest push --all cursed-image docker://nalind/cursed-image
```

Yes, you can do this. No, you shouldn't do this. And definitely don't try to use this as a base image for building other images.

Red Hat

Thank You!

buildah.io

https://github.com/nalind/devconfcz2020

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat