OCI Image Spec for VMs

The Open Container Initiative Image Specification includes specifications for how images should be constructed and stored. This document contains proposed changes to that specification to support proper VMs.

While following this proposal, we need to keep in mind the work of the OCI Artifacts initiative. That will solve the *packaging* question, but will not solve the *r untime* question. Specifically, the goal is to have an OCI-compatible runtime be able, optionally, to run both OS virtualized images and hardware virtualized images.

This is not yet ready for submission to OCI. It is intended as a working space until we have a proposal that can be submitted.

Current Spec

The current spec mandates that the root manifest of an image is one of the following two items:

- Index: An index of multiple images that can satisfy the requirements, based on certain characteristics, e.g. OS or architecture
- Manifest: An image itself, whose manifest consists of only a few key fields:
    - version - integer
    - media-type - string
    - config - object (technically, an OCI descriptor)
    - layers - array of objects, each an OCI descriptor
    - annotations - string key-value map

According to the image spec, the media-type of the manifest, as well as of each layer and the config, must be of a specific limited set of types.

The config itself has a defined schema and a limited set of fields, as defined by its media type `application/vnd.oci.image.config.v1+json`

Proposed changes

Limitations:

- The proposed changes do not affect an index, only a manifest.
- The proposed changes do not affect the layers or annotations of an index, only the config.

The proposed changes will add the following fields to the `config` section of the configuration document:

- bootable: boolean. OPTIONAL. Indicates if the image is an OS virtualized image, and thus should be run using the kernel of an existing OS, or hardware virtualized, i.e. a "traditional" VM image, and thus should be booted. Defaults to false.
- os: array of string. OPTIONAL when bootable is "true", PROHIBITED otherwise. An array of resources related to starting a container baremetal or inside a virtual machine, where a kernel is necessary and comes with the container. Entries are in the format `KEY=VALUE`.
    - firmware: string. OPTIONAL. Type of firmware to boot the system. May be "UEFI" or "BIOS". Defaults to "UEFI".
    - kernel: string. OPTIONAL. The path to the kernel binary to use as entry point when the container starts. If not present, boots the image as a boot disk using the firmware.
    - ramdisk: string. OPTIONAL. The path to the ramdisk to use with the kernel when the container starts.
    - cmdline: string. OPTIONAL. The kernel command line to use with the kernel when the container starts.
    - DTB: string. OPTIONAL. The path to the device tree binary to be loaded and passed to the kernel when the container starts.

All paths in the `os` section are relative to the root of the image.

The above should be sufficient to boot any image.

Additionally, it is possible that certain images may require specific hardware to be usable, for example, certain FPGAs or GPUs. To support these, we submit a separate proposal for resources, also under the `config` section of the configuration document:

- Resources: array of string. OPTIONAL. An array of hardware resources requested by the container to be accessible. Entries are key-value pairs of strings, in the format of `RESOURCE=DEVICE_DATA`, where: `RESOURCE` identifies a physical resource to be assigned to the container. Initial supported hardware identifiers are `PCI BDFs` and `device tree paths`. `DEVICE_DATA` is the binary file to be loaded into the hardware resource before assigning it to the container to make it usable. `null` specifies that no binaries need to be loaded. The mechanism to load the `DEVICE_DATA` binary is specific to the hardware resource in question.

For Resources, examples are:

- `00:02.0=/lib/firmware/ipu3-fw.bin`
- `/amba/pr@01=/boot/bitstream`