Utilizing Docker Hub in Docker build jobs

Why Docker Hub is recommended for Multi-Arch

Docker Hub registry is able to store manifest list (or fat manifests). A manifest list acts as a pointer to other images built for a specific architecture, thus making it possible to use the same name for images that are built on hardware with different architectures.

blocked URL

Figure 1: Docker registry storing amd64, arm64 images and their fat manifest

In the picture above *akraino/validation:k8s-latest* is the fat manifest, and its name can be used to reference both images *akraino/validation:k8s-amd64-latest* and *akraino/validation:k8s-arm64-latest*. Inspecting the manifest offers the details on what images it has, for what hardware architecture and what OS.

blocked URL

Figure 2: Docker fat manifest details

Custom job: validation

The validation project is already pushing to Docker Hub, so if you would like to check out some template code, please take a look at ci-management/jjb/validation.

The docker images are in the official repo now [1] and the docker build jobs are passing [2].

[1] https://hub.docker.com/r/akraino/validation/

[2] https://jenkins.akraino.org/view/validation/job/validation-master-docker/

Building multi-arch images on your project

In order to have a multi-arch pipeline in your project you need to:

1. Determine the hardware where you run your containers (for each architecture). You might want to setup your own or you can contact the lab owners from the Akraino community to share their resources. Right now Akraino supports x86 and aarch64 architectures
2. If you are using the current Akraino resources, you can skip this step. If you are using your own hardware resources, you need to setup your Jenkins slave. If you also want to connect the slave to LF master, contact LF at https://jira.linuxfoundation.org/servicedesk/customer/portal on how to make the connection.
3. Replicate the Jenkins jobs that are used for building multi-arch images. You can use the code here as a reference, which creates a multi-job that builds and pushes the images in parallel for each architecture, then create and push the fat manifest for those images. The code from the example is ran in LF CI infrastructure here, and the images are being pushed in this docker repository.