Docker Image Tags Handling, Image SHA usage in EVE

1 Introduction:

This document describes the way we handle a couple of issues:

1. Container Image Tags
2. Container Image SHA

2 Container Image Tags

For containers, for an image with tags, we need a way to make eve refresh the image. For example, if the image is ubuntu:latest, once we download the image, we do not attempt to re-download it though there is a new image for ubuntu:latest. We need a manual trigger for the user to make it download the image. Similar question arises when creating a second container on the same device with the same image. Should an AppInstance always check the image registry for the latest image? Or is it expected to use the already downloaded image (if available) - and user refreshes manually?

As a comparison, currently, For VMs, when the user changes the image, the controller triggers (optionally) an update to the App Instance Config.

The difference between VM and container is, for a VM, for the image to change, the ImageID needs to change. So Eve can detect the change in image. For container, unless it downloads the manifest again, it cannot be sure if there is a new version of the image or not.

Please note - this may change going forward, when VM images are also supported on docker registries.

2.1 App Instance Create

As EVE is used more and more in the cloud-native world, this is a practice that people are used to both from docker and kubernetes.

1. When the image doesn't exist, obviously it pulls
2. When the image does exist:
    a. Policy is set to "always pull":
    b. It checks for a new manifest, hashes it, and sees if that hash is different than the current one. If it is, it pulls the entire thing, else it does not (because the hashes prove it is the same)
    c. Policy is set to "use existing", it does not pull the manifest
    d. Policy is not set, use default

The definition of "exists" is that whatever is on disk matches the config. If the config was only "ubuntu:latest", then it is that tag. If the config was "ubuntu: latest@sha256:abcdef55511" (i.e. includes a hash), then it validates hash as well.

A similar functionality could be achieved using the following flags:

1. “ImagePullPolicy” in Device Config
    a. This will act as the default Policy for the device.
    b. The default “Default Policy” (device.ImagePullPolicy) will be “Always Pull”
2. “ImagePullPolicy” in App Instance Config
    a. This can be set to override device level policy and specify what needs to be done for the App Instance.
3. This ImagePullPolicy can get more detailed, to handle other scenarios than just AppInstance Create, like what to do in case of Image Pull Failure, Should we periodically pull the images, Should image pull happen on each reboot of device or Restart of App Instance etc..

We think at this point, we don’t need all those knobs and **just go with the default policy of “Always pull the image” for App Instance Create**. If the pull fails, the app instance will fail, even if there is an existing version of the image.

We will implement the ImagePullPolicy knob if there is a future need.

2.2 EVE API for Image Refresh

We propose we add a new field to AppInstanceConfig:

```
InstanceOpsCmd refreshImage = 21;
```

When this is set, EVE will try to the following:

1. Checks if a new image exists for each of the drives used by the App Instance. If yes, download the image.
2. If there is a new image for **ANY** of the drives, the app will be restarted.
    a. If there are no new images, the app is **NOT** restarted (if restart is also not set)
3. It DOES NOT affect non-drive images (HDD_UNKNOWN)
4. This refreshImage counter can be combined with the other 2 counters.
    a. If this refreshImage is set, the ModifyConfig, RestartCounter and PurgeCounter are deferred till the download attempt is complete
5. refreshImage + **Restart counter**
    a. If the *restart* counter is set along with *refreshImage* counter, the restart is deferred till all the image downloads are done. Once all the images are downloaded (no-op if no new images), then the App is restarted
    b. NOTE - if restart counter is NOT set, App is restarted only if there is a change in one of the images.

1. refreshImage + **Purge counter**
    a. If both *refreshImage* and *purge* counters are set, purge counter is deferred till refresh is done. Then the app is restarted with the latest versions of all the images, discarding any state accumulated by the app.

3 Container Image SHA

1. All VM IMages will have the sha specified for verification
    a. Current behavior - no changes here.
2. For Containers:
    a. In an Image, User can OPTIONALLY specify SHA of manifest.
        i. If specified, this is verified after the manifest is downloaded.
    b. If Manifest SHA is NOT Specified
        i. Download the manifest, Calculate the SHA and use it for internal purposes (Image Tracking).
        ii. Report the SHA back to the controller in info msg for app instance.
3. Need to add an abstraction from ImageObjectID to SHA of the Image.
    a. Currently, there is an assumption that IMAGE-OBJECT-ID is unchangeable. This needs to change.
        i. This may be true for VMs also, as we allow the VM image to be stored in a docker registry.
    b. In EVE, ImageObjectID needs to be resolved to a SHA
        i. Multiple images can resolve to the same SHA.
        ii. ImageObjectID -> SHA is resolved during App Instance Create
            1. This can be updated using the “RefreshImage” trigger.

4 APPENDIX - Comment from AVI - Alternate Proposal

AVI: I think there is a simpler solution. While you are right that the VM changes the ImageID and the container doesn’t, the container has a very lightweight manifest which is the source of the hash, and can be checked. Is there any reason we cannot, every time we start an ECO download the manifest and check its hash against whatever we have? The manifest typically is a few hundred bytes. If the manifest has changed, we invalidate the old and get a new one; if it has not, we do not.

The work Erik is doing around using the hash in verified will make this easier.

The remaining question then becomes, when does a user *force* us to check? That only would play in if there is no ECO starting up, and the user wants to force a replacement.

Kalyan’s Response:

The issue is - User should have control over that behavior. Remember this is not data center - It may not always have connectivity. It may not have the bandwidth to download the container.So we need the user to explicitly tell us if they want to upgrade the image.

Remember - these changing tags - are mostly a developer workflow.. Not the end deployment one. In end deployments, they will likely point to a non-mutable tag.

As you said - if we indeed want to provide a mode to make the container check on startup every time, I think we should make it a separate option.. not not always do it.