Tags Matter

In a previous lesson, we mentioned that an image name can include an optional tag. In this lesson, we will discuss tags in a bit more detail and learn why they are important.

WE'LL COVER THE FOLLOWING

Earlier, we saw that image names include a name and a tag. As a quick reminder, an image name is:

```
<repository_name>/<name>:<tag>
```

- *tag* is optional; when missing, it is considered to be *latest* by default
- *repository_name* can be a registry DNS or the name of a registry in the Docker Hub

While your images aren't published to a registry (at least in this chapter, but that will change in the next one), you don't need to include a registry name. So, your image name is:

```
<name>:<tag>
```

The *latest* tag

In my demonstrations I didn't include a tag; therefore the default *latest* tag was used. For instance, the actual image name was *hello:latest* when I ran the following command:

```
docker build -t hello .
```

As long as you are creating simple software, running on a simple CI/CD pipeline, it can be fine to use the *latest* tag. In a simple scenario, you may:

1. Update the source code
2. Build a new image with the *latest* tag
3. Run a new container with the newest image
4. Kill the previous container

There's a caveat with this however: when using the *docker run hello* command on a distant machine (which actually means docker run *hello:**latest***), the distant machine has no means to know that there is a newer version of the *hello:latest* image. You need to run the *docker pull hello* command on the distant machine in order for the newest version of your image to be downloaded to that machine.

This may sound awkward, and that's one reason for not just using the *latest* tag.

Why Would You Tag Your Images?

Other reasons come to mind once you become more serious with your CI/CD pipeline. For instance, you may want any or all of the following features:

- Be able to roll back to a previous version of an image if you detect a problem with the latest image.
- Run different versions in different environments. For instance, the latest version in a test environment and the previous version in a production environment.
- Run different versions at the same time, routing some users to the latest version and some to the previous versions. This is known as a canary release.
- Deploy different versions to different users, and be able to run whatever version on your development machine while you support them.

These are all good reasons for tagging your images. If you ensure each released image has a different tag, you can run any of the scenarios

mentioned above.

You're free to tag your images however you want. Common tags include:

- a version number, e.g. *hello:1.0*, *hello:1.1*, *hello:1.2*
- a Git commit tag, e.g. *hello:2cd7e376*, *hello:b43a14bb*

In order to apply a tag, just state it during your build command:

```
docker build -t hello:1.0 .
```

Tags for Base Images

Remember your images are based on other images; this is done using the *FROM* instruction in your *Dockerfile* file. Just as you can tag your images, the base image you use can be the *latest* one or a tagged one.

In my demos, I used tagged images. For instance, I based my server image on the *nginx:1.15* base image. It's quite tempting to base your images on the latest ones so that you're always running on up-to-date software, especially since it's so straightforward. All you need to do is omit the tag altogether or mention the *latest* one. You could be tempted to use the following instruction in your *Dockerfile* file:

```
FROM nginx:latest
```

Don't! First of all, it doesn't mean that any running container will be based on the latest available version of the *nginx* image. Docker is about having reproducible images, so the *latest* version is evaluated when you build your image, not when the container is run. This means that the version will not change unless you run the *docker build* command again.

Second, you're likely to run into trouble. What about the *nginx* image releasing a new version with breaking changes? If you build your image again, you're likely to get a broken image.

For these reasons, I recommend specifying the image tag. If you want to keep up to date with new releases of the base image, update the tag manually and make sure you test your image before releasing it.

make sure you test your image before releasing it.

In the next lesson, we will look at *environment variables*.