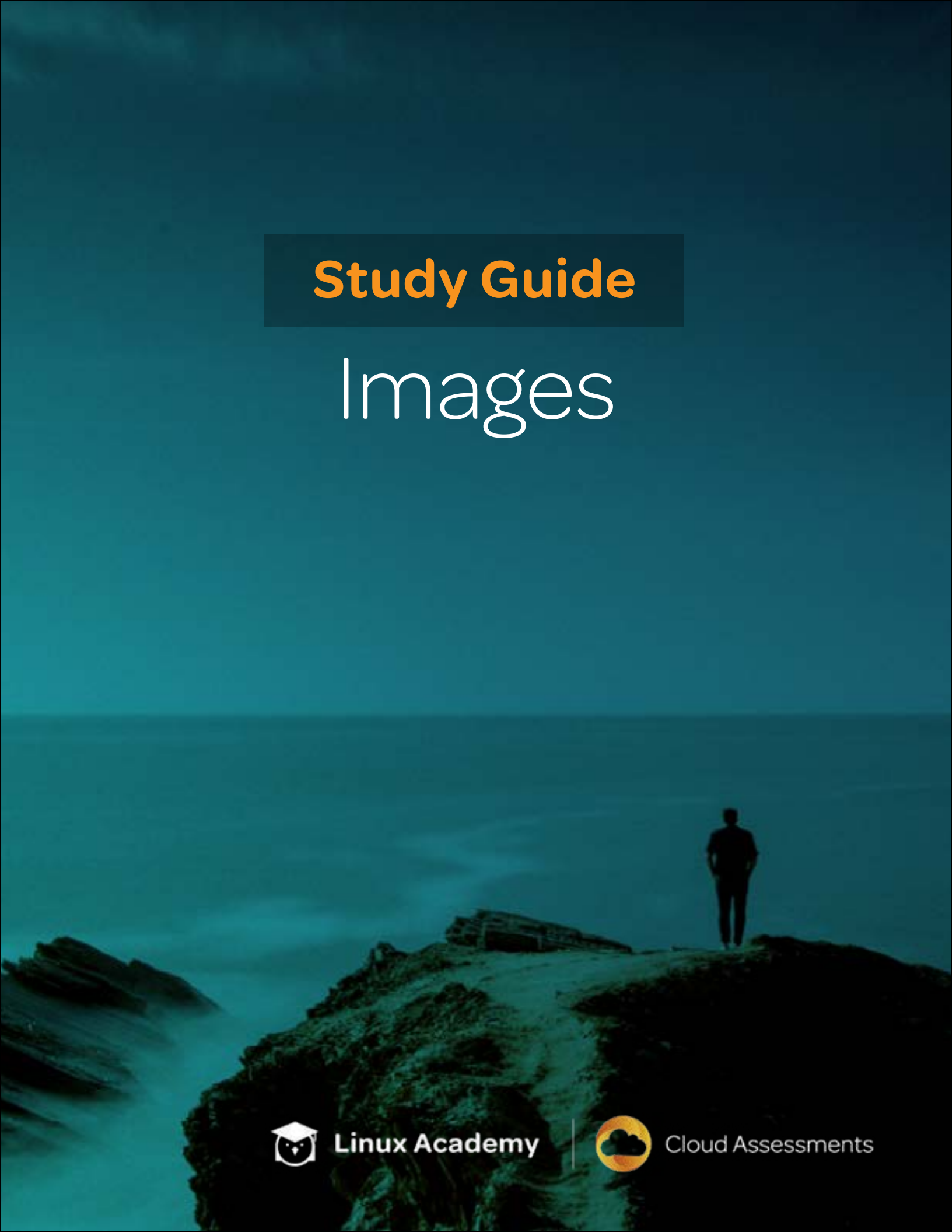
Study Guide

Images



Linux Academy | Cloud Assessments

Contents

This guide accompanies the Docker Certified Associate Prep Course and reviews how to:

- Pull an image from a registry.
- Search for an image in a repository.
- Tag an image.
- Use CLI Commands to manage images (list, delete, prune, RMI, etc.).
- Inspect images and report specific attributes using the flags –filter and –format.

Reviewing Images

In these sections, we will go over how to pull, view, and search for images in DockerCE.

Pull an Image

Similar to Git, DockerCE uses Docker hub for its registry. There are a few ways to pull an image into DockerCE:

- Pull a repository image using docker pull. We are using the file hello-world as our example:

```
docker pull hello-world
```

By default, docker pull pulls a single image, though repositories can have multiple versions of an image.

- Pull multiple versions of an image using the -a or the –all-tags option with docker pull:

```
docker pull -a hello-world
```

- Pull down an image without verifying that it has been signed by the repository using the –disable-content-trust command. Note that this command can be extremely dangerous as we are trusting an unverified image:

```
docker pull -disable-content-trust hello-world
```

- Pull an image using a specific version using the docker pull command. For this example, we are pulling centos:6:

```
docker pull centos:6
```

This will pull only the centos file with the image tag of 6.

```
[user@ellmarquez1 ~]$ docker images REPOSITORY TAG IMAGE ID
CREATED SIZE centos 6 b5e5ffb5cdea 2 weeks ago 194MB hello-
world latest 2cb0d9787c4d 6 weeks ago 1.85kB hello-world linux
2cb0d9787c4d 6 weeks ago 1.85kB Note: These are the images
currently on the system used for this guide; you may see something
different.
```

View Images on Our Server

The following are ways to view different images on our server:

- Use the docker images command to view all current images on the Docker system:

```
[user@ellmarquez1 ~]$ docker images REPOSITORY TAG IMAGE ID
CREATED SIZE hello-world latest 2cb0d9787c4d 6 weeks ago 1.85kB
hello-world linux 2cb0d9787c4d 6 weeks ago 1.85kB
```

- Above, we see the REPOSITORY, TAG, short number IMAGE ID, when the image was created, and the size of the image. If the full IMAGE ID is needed, use the command docker images –digest:

```
[user@ellmarquez1 ~]$ docker images -digests REPOSITORY
TAG DIGEST IMAGE ID CREATED SIZE hello-world latest
sha256:4b8ff392a12ed9ea17784bd3c9a 6 weeks ago 1.85kB
```

- We can also use filters to find what images existed on the system before a certain point using docker images –filter "before=<image name>". For our example, we are using centos:6:

```
[user@ellmarquez1 ~]$ docker images -filter "before=centos:6"
REPOSITORY TAG IMAGE ID CREATED SIZE hello-world latest
2cb0d9787c4d 6 weeks ago 1.85kB hello-world linux 2cb0d9787c4d 6
weeks ago 1.85kB
```

Quick Hack: Use docker images -q to get a list of all of the IMAGE ID numbers on your system. This can be helpful if we need an image ID to pass to another command.

Searching for Images

The following are ways to search for and review images:

- To perform a base search for an image, use the docker search <image name> command. For example, let's search for the apache image.

```
docker search apache
```

We get back multipeapache images.

- To narrow down the results, we can use the –filter flag to search for specific forms of the image. For this example, we'll search for only official images of apache using –filter is-official=true:

```
docker search -filter is-official=true apache
```

Despite filtering down, we may still have quite a few options.

- Let's narrow down the search one more time. For this search, let's only look for official images that have a rating of over 50 stars using stars=50. Remember, each –filter flag can only pass one option, so we will need two filters to search for both:

```
docker search -filter stars=50 -filter is-official=true apache
```

Image Tags

Image tags are used to help find files that we may not know the name of, but we know their tags. Lets take another look at our images and see which ones have tags:

```
[user@ellmarquez1 ~]$ docker images REPOSITORY TAG IMAGE ID
CREATED SIZE centos 6 b5e5ffb5cdea 2 weeks ago 194MB hello-
world latest 2cb0d9787c4d 6 weeks ago 1.85kB hello-world linux
2cb0d9787c4d 6 weeks ago 1.85kB
```

We can see above a section labeled TAG. In order to tag an image, we use the docker tag command. Let's try creating a tag for the centos image and give the image the name my centos:

```
docker tag centos:6 mycentos:1
```

Let's check to make sure our tags worked correctly using the docker images command:

```
[user@ellmarquez1 ~]$ docker images REPOSITORY TAG IMAGE ID
CREATED SIZE centos 6 b5e5ffb5cdea 2 weeks ago 194MB mycentos 1
b5e5ffb5cdea 2 weeks ago 194MB hello-world latest 2cb0d9787c4d 6
weeks ago 1.85kB hello-world linux 2cb0d9787c4d 6 weeks ago 1.85kB
```

Notice that mycentos and centos have the same image ID. This is because all we did was create a copy of the centos image that we could later use to make our own custom image without affecting the original centos image.

Note: If we plan on storing this image locally, using the source centos and tag 6 would suffice. However, if we were planning on sharing this image on a registry, we would need to use the command <registery>/<image name:<tags>. For example, docker tag centos:6 myreg/mycentos:2.

Docker Image Commands

The following are other commands you can use with Docker images:

- To view the docker history, we use the docker history command. Docker history lets us see the layers that compose the image. For this example, we'll look at the history for mycentos:1:

```
[user@ellmarquez1 ~]$ docker history mycentos:1 IMAGE CREATED
CREATED BY SIZE COMMENT b5e5ffb5cdea 2 weeks ago /bin/sh -c
#(nop) CMD ["/bin/bash"] 0B <> 2 weeks ago /bin/sh -c #(nop) LABEL
org.label-schema.sc… 0B <> 2 weeks ago /bin/sh -c #(nop) ADD
file:769078df784180af4… 194MB
```

- Let's remove the hello-world image with the linux tag. We can do this in one of two ways. The first is we use the rm command:

```
docker image rm hello-world:linux
```

Or we use the rmi command:

```
docker rmi hello-world:linux
```

- Since we have two versions of hello-world, one with the linux tag and one with the latest tag, the hello-world image designated is not removed, only untagged:

```
[user@ellmarquez1 ~]$ docker rmi hello-world:linux Untagged:
hello-world:linux -
```

To make sure the image was removed, use docker images. Note that hello-world:latest is still in the file system, the only thing removed was the linux tag.

```
[user@ellmarquez1 ~]$ docker images REPOSITORY TAG IMAGE ID
CREATED SIZE centos 6 b5e5ffb5cdea 2 weeks ago 194MB mycentos 1
b5e5ffb5cdea 2 weeks ago 194MB hello-world latest 2cb0d9787c4d 6
weeks ago 1.85kB
```

- Save your docker image using the docker image save command along with the name of the image, the image's tag, and what you want to save it as. For our example, we are saving the mycentos:1 image as mycentos.tar:

```
docker image save mycentos:1 > mycentos.tar
```

- The image and the underlying filesystem layer, along with the meta data, will be saved. This image can now be transferred to a new system. We can also check to make sure it saved correctly by listing out all files of that type. As we did for our example above, we want to list out .tar files:

```
[user@ellmarquez1 ~]$ ls *.tar mycentos.tar
```

- Import a docker image with docker import, followed by the file type, repository, and the tags<file> <repository:tag>. For our example, we're importing the mycentos.tar file from the localimpoart repository with the centos6 tag:

```
[user@ellmarquez2 ~]$ docker import mycentos.tar
localimport:centos6 sha256:65c5dff95dd1c7e69081078 [user@
ellmarquez1 ~]$ docker images REPOSITORY TAG IMAGE ID CREATED SIZE
localimport centos6 65c5dff95dd1 4 seconds ago 202MB
```

Note: docker load will load the image without taking an argument for a name using the defaults:

```
docker load -input mycentos.tar
```

- Docker Prune command docker image prune will remove all "dangling" images. These are images that are not currently associated with a complete image or a container. By adding the -a flag, we remove all images not associated with a container:

```
[user@ellmarquez1 ~]$ docker image prune -a WARNING! This will
remove all images without at least one container associated to
them. Are you sure you want to continue? [y/N]
```

Inspecting Images

To inspect an image, use the command docker image inspect. However, you may find that it becomes easier to look through this information if you redirect the info to a file. For our example, we will send it to centos.output:

```
docker image inspect > centos.output
```

If you are looking for specific information, the –format flag can help. For example, if we wanted the hostname associated with our centos image, we use the –format tag. For our example, we are using ContaienerConfig, which would be the top level section, and Hostname, which is where we pulling our information from:

```
[user@ellmarquez1 ~]$ docker image inspect centos:6 -format '{{.
ContainerConfig.Hostname}}' f185c8f40489
```

For clarity, here is the relevant section of the docker image inspect output:

```
"Parent": "",
 "Comment":"",
 "Created":"2018-08-06T19:22:45.144404666Z","Container":
 "f185c8f40489cf4921d51714053a",
 "ContainerConfig": {
      "Hostname":"f185c8f40489",
      "Domainname":"",
      "User":"",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
           "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
           bin:/sbin:/bin"
      ],
```

If we want the entire section, we can request it in .json format using –format '{{json .ContainerConfig}}':

```
[user@ellmarquez1 ~]$ docker image inspect centos:6 -format
'{{json .ContainerConfig}}'
```

All information for this image will appaer.