# More About Docker Run

In this lesson, you will further explore the 'docker run' command by running the 'alpine' image.

You can think of the *docker run* command as the equivalent of buying a new computer, executing some command on it, then throwing it away. Each time a container is created from an image, you get a new isolated and virgin environment to play with inside that container.

What you get inside the container depends on which image your container is based on. After the image name, you can pass the commands you want to execute inside the container.

Let's illustrate the new-computer-that-you-trash metaphor using the *alpine* image. The *alpine* image is a very small Linux image that does enough for our purpose.

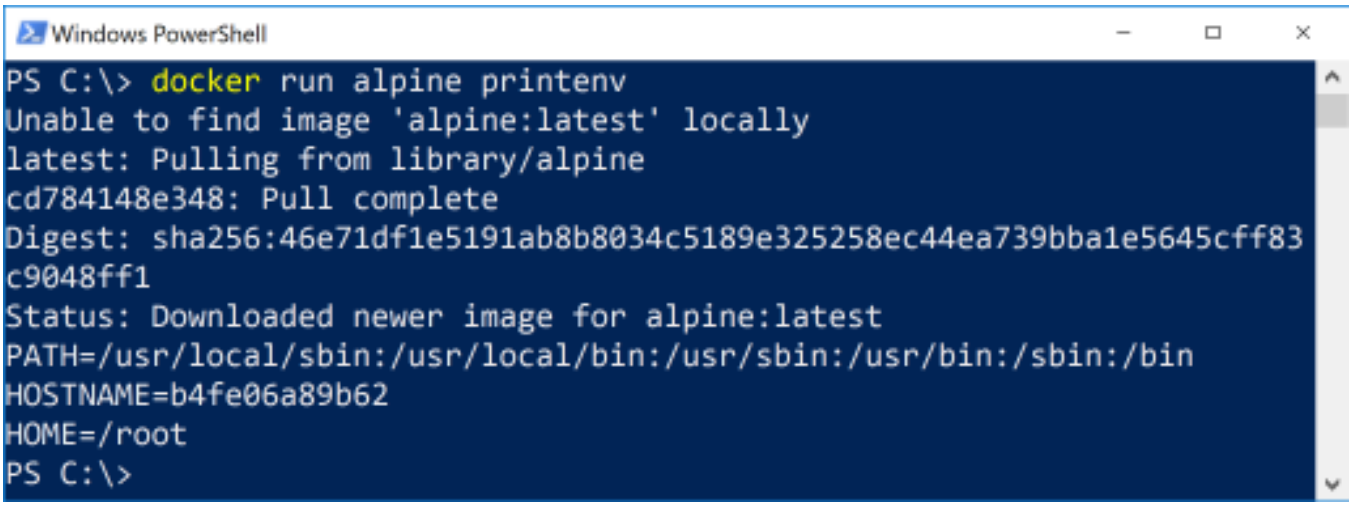
Let's run a container and ask it to display its hostname.

```
docker run alpine printenv
```

We are basically asking for a container to be created using the *alpine* image, and for the container to execute the *printenv* command that is one of the binary programs packed in the *alpine* image.

To no surprise, the *alpine* image is downloaded in order to create the container, since it was not already present on my disk.

The result is the following:

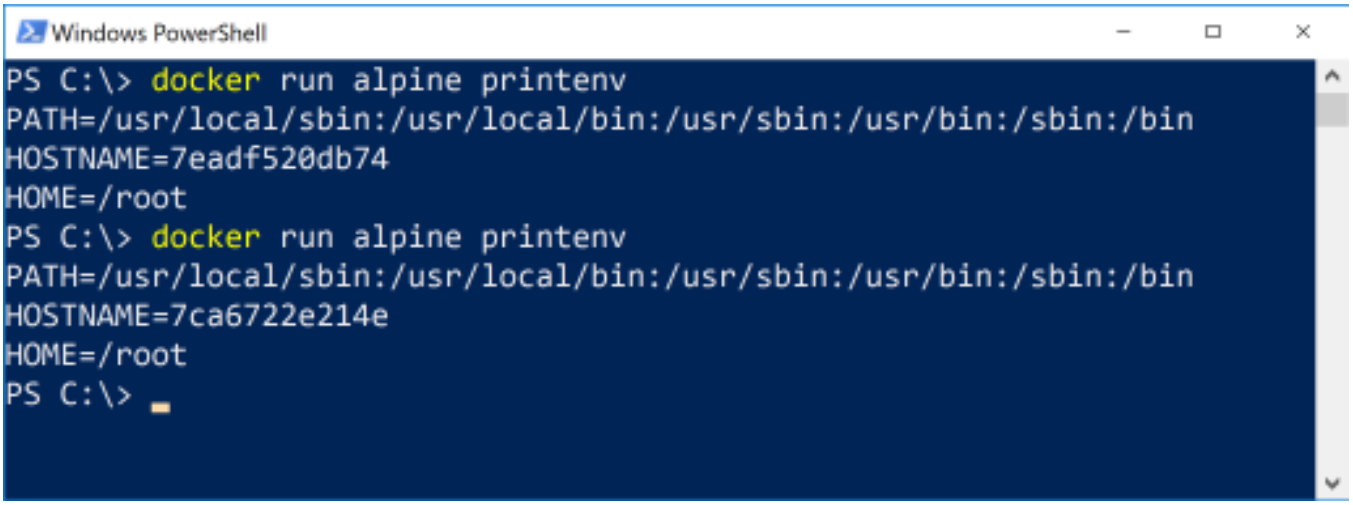



Now let's run two more containers and ask them to display their respective hostnames.

```
docker run alpine printenv
docker run alpine printenv
```

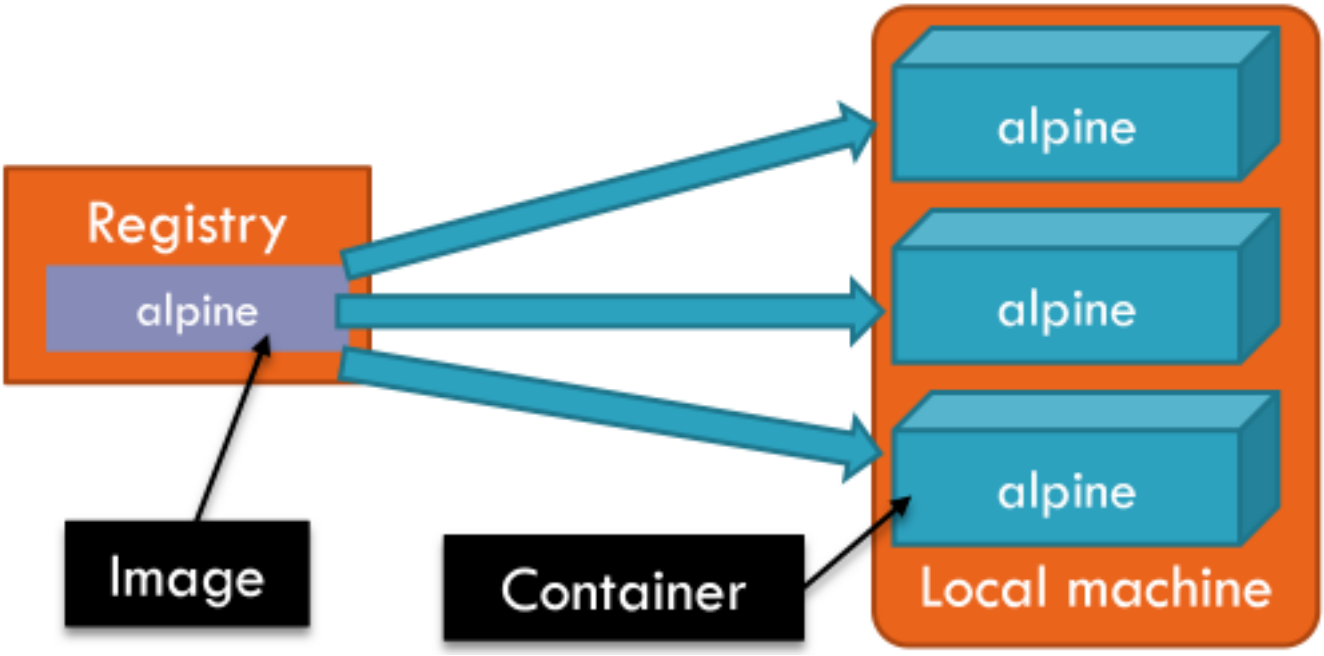
This is the same command as above, repeated twice. On each line we're asking for a new container to be created using the *alpine* image, and for the container to execute the *printenv* command.

The *alpine* image isn't downloaded because it is already on our disk. Two more independent containers are created. The result is the following:




Note that inside each container the displayed *hostname* is different. See? You get the equivalent of a new machine on each container.

Here is a simplified schema of what I did:




Creating three independent containers using the alpine image

What I want you to understand is that Docker is a tool that allows you to get the equivalent of a disposable, single-time use computer. Once that's clear, a whole new world opens for you. You come from a world where obtaining a new machine and configuring it required enough efforts to justify keeping it, despite all the side effects that come with each subsequent use. In the container's world, getting a brand-new environment is cheap enough for it to get many of them.

If I list the stopped containers on my machine (docker ps -a), I'll get 3 stopped containers since I just asked for that number of containers to be created. Being quite lazy, I don't want to remove them one by one, so I use the following handy command:

```
docker container prune -f
```

This is the equivalent of running one *docker rm* command for each stopped container. The *-f* switch is an implicit confirmation to proceed and delete all stopped containers right away, instead of asking to confirm that operation.

---

In the next lesson, we will take a look at long-run containers.