Parameters as Environment Variables

In this lesson, you will be introduced to environment variables in the context of containers and learn how to set environment variables in a container.

WE'LL COVER THE FOLLOWING

In real life, a container's inputs and outputs are likely to vary according to the container's environment. For instance, if you run a web application, it is likely to connect to a database and listen for incoming requests on a given DNS. The database connection details and DNS will have different values on a development machine, on the test server, and the production server.

Reading a Value

Whatever the technology you use inside your container, you can access environment variables. For instance, if you set a *name* environment variable, you may access it with:

Technology	Access
Linux shell	$name
.NET Core	.AddEnvironmentVariables();
Java	System.getenv("name")

Node.JS	process.env.name
PHP	.$_ENV[“name”]
Python	os.environ.get(‘name’)

Providing a Value

On a real machine, environment variables are set on your system. Inside a container, they can be set from several sources, which make them appropriate for parameterizing your containers.

In order to provide an environment variable’s value at runtime, you simply use the *-e name=value* parameter on the *docker run* command.

> A special use case is when the system that runs the container has the *name* environment variable defined, and you want to reuse it, then you can simply use the *-e name* parameter without specifying a value.

Default Value

You may also want to define a default value for an environment variable, in case it isn’t provided when a container is created; this may be done in the *Dockerfile* file, using the *ENV* instruction. For instance, the following makes sure that if the *name* variable isn’t provided to the *docker run* command, it has a default value of *Dockie*:

```
ENV name=Dockie
```

It’s good practice to add an *ENV* instruction for every environment variable your image expects since it documents your image.

Sample Usage

I want to create an image that can ping any given site. I’ll do this using a Linux shell script. I define it in a *ping.sh* file:

ping.sh

```
#!/bin/sh

echo "Pinging $host..."
ping -c 5 $host
```

Note that I make use of a *host* environment variable. I'm going to define an image that includes and runs that script:

Dockerfile

```
FROM debian:8

ENV host=www.google.com

COPY ping.sh .

CMD ["sh", "ping.sh"]
```

Note that my *Dockerfile* file includes an *ENV* instruction that specifies that the *host* variable will be *www.google.com* in case it isn't provided. I create my image from that *Dockerfile* file by running a *docker build* command:

```
docker build -t pinger .
```

Next, I run two containers based on that image:

```
docker run --rm pinger
docker run --rm -e host=www.bing.com pinger
```

We don't provide the first container with any value for the *host* environment variable in order for it to default to the *www.google.com* value specified in the *Dockerfile* file. The second container is provided the *www.bing.com* value. Here's the output from these two containers (shortened for brevity):

```
C:\>docker run --rm pinger
```

```
Pinging www.google.com...
PING www.google.com (172.217.18.196) 56(84) bytes of data.

64 bytes from par10s38-in-f4.1e100.net (172.217.18.196): icmp_seq=1 ttl=37
 time=6.52 ms
[...]
64 bytes from par10s38-in-f4.1e100.net (172.217.18.196): icmp_seq=5 ttl=37
 time=7.35 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 5.632/7.854/11.031/1.890 ms

C:\>docker run --rm -e host=www.bing.com pinger
Pinging www.bing.com...
PING dual-a-0001.a-msedge.net (204.79.197.200) 56(84) bytes of data.
64 bytes from a-0001.a-msedge.net (204.79.197.200): icmp_seq=1 ttl=37 time
=7.82 ms
[...]
64 bytes from a-0001.a-msedge.net (204.79.197.200): icmp_seq=5 ttl=37 time
=8.08 ms

--- dual-a-0001.a-msedge.net ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 7.820/8.669/10.193/0.954 ms
```

You can see that each container pinged a different host, according to the values provided to them.

This was a simple demo, but you can provide advanced values. You would typically provide full connection strings or URLs to other services, usernames, and passwords, to name a few. This is a flexible and powerful feature since those values may come from many sources once you begin to use orchestrators.

Before we move on to storage, try the exercise in the next lesson.