**Cloud 300**

Optum

Student Exercise Manual

# Lab 2.1: Pulling a Container

## Overview

When we run a docker container, we very rarely ever start from scratch. Docker provides us an OS kernel (such as linux), but generally does **not** provide us with any of the other aspects of the OS we think of.

## Step 1: Do a Docker Pull

Docker pull will pull down a public repo from dockerhub. Think of it like a git pull.

We are going to use a lightweight linux container called [alpine](https://hub.docker.com/r/_/alpine/). You can read more about it [here](https://hub.docker.com/r/_/alpine/). It is only 4MB in size, making it perfect for a container.

docker pull alpine

You should get the results:

Using default tag: latest  
latest: Pulling from library/alpine  
ff3a5c916c92: Pull complete  
Digest: sha256:7df6db5aa61ae9480f52f0b3a06a140ab98d427f86d8d5de0bedab9b8df6b1c0  
Status: Downloaded newer image for alpine:latest

## Step 2: See the docker image we downloaded

REPOSITORY TAG IMAGE ID CREATED SIZE  
hello-world latest e38bc07ac18e 5 weeks ago 1.85kB  
alpine latest 3fd9065eaf02 4 months ago 4.15MB

There you see it. Alpine. But what do we do with it?

## Step 3: Run the Container

How do we run the container?

docker container run alpine ls

And we see what we ran: as we are in the root "/" directory, we see the directories there.

bin  
dev  
etc  
home  
lib  
media  
mnt  
proc  
root  
run  
sbin  
srv  
sys  
tmp  
usr  
var

## Step 4: Listing Containers

Let us try to list our container.

docker container ls

And we get the response:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**Wait??** where's our container? Well, our container isn't running at the moment. The command we gave it (ls) ended already. We can see the stopped container if we want like this.

docker container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
d060f9a50c7c alpine "ls" About a minute ago Exited (0) About a minute ago admiring\_keldysh  
887030cdbd7e hello-world "/hello" 26 minutes ago Exited (0) 26 minutes ago nostalgic\_lewin

There we go. We have two containers, both stopped. Remember hello world? That's there. Just stopped.

Wait, should we keep creating all these containers? You can stop it from being created if you just want to one-and-done a command.

docker container --rm ls -l

\*\*=> TODO: See if what containers we have. Does the new one show up?

## Step 5: Interactive shells

Wait, wow do I ssh to my container? Well, you don't usually do it that way. For one thing, your container isn't actually running right now, so if you try to ssh it won't respond. What you probably want is an interactive shell.

How do I do that?

docker container run -it --rm alpine /bin/ash

What does this mean? \* -i : interactive mode \* -t : terminal mode \* --rm : remove container after we are done \* /bin/ash : bash is big and needs to be installed. ash (almquist shell) is small. We also have old-school sh.

Here is the results with a few commands:

/ # echo hello  
hello  
/ # cd  
~ # pwd  
/root  
~ # echo bye!  
bye!  
~ # exit

**=> TODO: You try some of your own commands.**

## Summary

So what's the point? That we can run a mini size linux? Well, we are about to see how we can build on top of this. But usually we need to start somewhere. And we need some basic place to start.

# Lab 2.2: Manipulating a Container.

We are going to see how we can manipulate our containers

## Step 1: Start up a new alpine container

docker container run -it alpine

You will find yourself a root prompt in the new container. Try executing the command "ls /"

# ls /

bin dev etc home lib media mnt proc   
root run sbin srv sys tmp usr var

## Step 2: Delete all files in the container

Now let's delete all files in the container. This is very dangerous... you must make sure you are doing this **IN THE CONTAINER**, and not on a root prompt on your local system or your cloud VM.

# rm -rf /

**DO NOT TRY THIS AT HOME!!**

Now let's see if the files are gone

# ls

/bin/sh ls: not found

Everything's gone. You've trashed your system. Congradulations!

**=> Exit out of your container by hitting control-D**

Now, let's see what happens if we run our container again.

docker container run -it alpine

Ok, let's see if our files are there

ls -l

bin dev etc home lib media mnt proc   
root run sbin srv sys tmp usr var

=>\*\* Why are all the files back after we deleted them all?\*\*

Go ahead and exit by hitting CNTRL-D or typing exit.

## Step 3: Deleting Containers

Now, see if you can see your command:

docker container ls -as

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES SIZE  
c183852a7215 alpine "/bin/sh" 6 minutes ago Exited (0) 2 minutes ago adoring\_joliot 5B (virtual 4.15MB)

Notice the size. 5 bytes. Not exactly taking up a lot of space. The container IMAGE takes up 4.15MB.

But, we may want to clean up our images anyway. We can use docker container rm for that.

docker container rm <container-id>

And you can now run docker container ls -as

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES SIZE

It shouldn't be there anymore.

What if you want to run the docker container and just have it auto-delete after you are done.