RX-M

Docker Containers

Lab 4 – Power User Container Features

In this lab you will get a chance to do more advanced Docker container management.

1. Docker inspect

The `docker inspect` subcommand provides access to all of the meta data associated with a container. To experiment with inspect we'll create a container with several known bits of meta data, like a name and labels. Create the following container:

```
user@ubuntu:~$ docker create --name=websvr5 --hostname=ws05 --label="copyright=2016" --
label="lic=apache2" httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd

357ea8c3d80b: Pull complete
c4db84bae208: Pull complete
30f260b31041: Pull complete
3042e77c51ef: Pull complete
99842fac4df9: Pull complete
Digest: sha256:619b6136504fcaed0d1e1f9fac57cf69eedd174f98889d5b727598ca658d0bfc
Status: Downloaded newer image for httpd:latest
ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b
```

Docker stores the metadata for containers in it operating directory under a subdirectory called "containers". Locate docker's operating directory:

```
user@ubuntu:~$ docker info 2>&1 | grep -i "docker root"
Docker Root Dir: /var/lib/docker
```

You can simply run `docker info` if you like but the above `grep` command selects just the output we need, the Docker root directory. This directory offers only limited access to non root users. Become root and list the contents of the Docker root, locate the container metadata directory and display its contents:

```
user@ubuntu:~$ sudo su
[sudo] password for user:

root@ubuntu:/home/user# ls -l /var/lib/docker
total 32
drwx------ 5 root root 4096 Aug 21 13:34 aufs
drwx------ 3 root root 4096 Aug 21 23:22 containers
drwx------ 3 root root 4096 Aug 21 13:34 image
drwxr-x--- 3 root root 4096 Aug 21 13:34 network
drwx------ 2 root root 4096 Aug 21 13:34 swarm
drwx------ 2 root root 4096 Aug 21 23:22 tmp
drwx------ 2 root root 4096 Aug 21 13:34 trust
drwx------ 2 root root 4096 Aug 21 13:34 volumes
```

```
root@ubuntu:/home/user# ls -l /var/lib/docker/containers/
total 4
drwx------ 2 root root 4096 Aug 21 23:22
ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b

root@ubuntu:/home/user#
```

Docker creates a subdirectory for each container created under the containers directory. List the contents of the containers directory:

```
root@ubuntu:/home/user# ls -l
/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/
total 8
-rw-rw-rw- 1 root root 1886 Aug 21 23:22 config.v2.json
-rw-rw-rw- 1 root root 1059 Aug 21 23:22 hostconfig.json
```

This listing shows two files. One is the host independent configuration for our container (config.v2.json) and the other is the host dependent configuration file (hostconfig.json.) The host config contains things that might change from host to host. Docker containers attempts to be as portable as possible so most of the interesting information is in the config.v2.json. List both files:

```
root@ubuntu:/home/user# cat
/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/config.v2.j
son

{"State":
{"Running":false,"Paused":false,"Restarting":false,"OOMKilled":false,"RemovalInProgress":false,"Dead":f
alse,"Pid":0,"StartedAt":"0001-01-01T00:00:00Z","FinishedAt":"0001-01-
01T00:00:00Z","Health":null},"ID":"ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b","C
reated":"2016-08-22T06:22:05.684955354Z","Managed":false,"Path":"httpd-foreground","Args":[],"Config":
{"Hostname":"ws05","Domainname":"","User":"","AttachStdin":false,"AttachStdout":true,"AttachStderr":tru
e,"ExposedPorts":{"80/tcp":{}},"Tty":false,"OpenStdin":false,"StdinOnce":false,"Env":
["PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin","HTTPD_PREF
IX=/usr/local/apache2","HTTPD_VERSION=2.4.23","HTTPD_SHA1=5101be34ac4a509b245adb70a56690a84fcc4e7f","HT
TPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.23.tar.bz2"],"Cmd":["httpd-
foreground"],"Image":"httpd","Volumes":null,"WorkingDir":"/usr/local/apache2","Entrypoint":null,"OnBuil
d":null,"Labels":
{"copyright":"2016","lic":"apache2"}},"Image":"sha256:462fca6e7913e7758cbc2b476e61b78670bc9ab2573aeb3f6
b22dcd15af32f15","NetworkSettings":
{"Bridge":"","SandboxID":"","HairpinMode":false,"LinkLocalIPv6Address":"","LinkLocalIPv6PrefixLen":0,"N
etworks":{"bridge":
{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"","EndpointID":"","Gateway":"","IPAddress":
"","IPPrefixLen":0,"IPv6Gateway":"","GlobalIPv6Address":"","GlobalIPv6PrefixLen":0,"MacAddress":""}},"S
ervice":null,"Ports":null,"SandboxKey":"","SecondaryIPAddresses":null,"SecondaryIPv6Addresses":null,"Is
AnonymousEndpoint":false},"LogPath":"","Name":"/websvr5","Driver":"aufs","MountLabel":"","ProcessLabel"
:"","RestartCount":0,"HasBeenStartedBefore":false,"HasBeenManuallyStopped":false,"MountPoints":
{},"AppArmorProfile":"","HostnamePath":"","HostsPath":"","ShmPath":"","ResolvConfPath":"","SeccompProfi
le":"","NoNewPrivileges":false}

root@ubuntu:/home/user# cat
/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/hostconfig.
json
{"Binds":null,"ContainerIDFile":"","LogConfig":{"Type":"json-file","Config":
{}},"NetworkMode":"default","PortBindings":{},"RestartPolicy":
{"Name":"no","MaximumRetryCount":0},"AutoRemove":false,"VolumeDriver":"","VolumesFrom":null,"CapAdd":nu
ll,"CapDrop":null,"Dns":[],"DnsOptions":[],"DnsSearch":
[],"ExtraHosts":null,"GroupAdd":null,"IpcMode":"","Cgroup":"","Links":
```

```
[],"OomScoreAdj":0,"PidMode":"","Privileged":false,"PublishAllPorts":false,"ReadonlyRootfs":false,"Secu
rityOpt":null,"UTSMode":"","UsernsMode":"","ShmSize":67108864,"Runtime":"runc","ConsoleSize":
[0,0],"Isolation":"","CpuShares":0,"Memory":0,"CgroupParent":"","BlkioWeight":0,"BlkioWeightDevice":nul
l,"BlkioDeviceReadBps":null,"BlkioDeviceWriteBps":null,"BlkioDeviceReadIOps":null,"BlkioDeviceWriteIOps
":null,"CpuPeriod":0,"CpuQuota":0,"CpusetCpus":"","CpusetMems":"","Devices":
[],"DiskQuota":0,"KernelMemory":0,"MemoryReservation":0,"MemorySwap":0,"MemorySwappiness":-1,"OomKillDi
sable":false,"PidsLimit":0,"Ulimits":null,"CpuCount":0,"CpuPercent":0,"IOMaximumIOps":0,"IOMaximumBandw
idth":0}

root@ubuntu:/home/user#
```

Notice that the output in both cases is valid JSON but for compactness it is all one line. This means that if you use grep to find things it wont help much, grep will output the entire line for every match.

- Consider your copyright label from the container in question, which file do you think it is in? Why?
- Do you think you will find a process ID (PID) in the metadata? Why or why not?

Next lets start the container we created and see what happens in the container's directory:

```
root@ubuntu:/home/user# docker start ac8a053
ac8a053
root@ubuntu:/home/user# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
ac8a05310b79        httpd               "httpd-foreground"   18 minutes ago      Up 29 seconds
80/tcp              websvr5
```

Now re-list the container directory:

```
root@ubuntu:/home/user# ls -l
/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/
total 28
-rw-r----- 1 root root  893 Aug 21 23:40
ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b-json.log
-rw-rw-rw- 1 root root 2734 Aug 21 23:40 config.v2.json
-rw-rw-rw- 1 root root 1059 Aug 21 23:40 hostconfig.json
-rw-r--r-- 1 root root    5 Aug 21 23:40 hostname
-rw-r--r-- 1 root root  166 Aug 21 23:40 hosts
-rw-r--r-- 1 root root  195 Aug 21 23:40 resolv.conf
-rw-r--r-- 1 root root   71 Aug 21 23:40 resolv.conf.hash
drwxrwxrwt 2 root root   40 Aug 21 23:40 shm
```

To start our container, Docker needs to create more support files. The .log file contains the log output of the container, this allows us to recover the container's log data even after it has stopped. The hostname, hosts, and resolv.conf are all standard networking files the Docker daemon will place in the container's etc directory. Note that if we started this image (httpd) 5 times, each container, while otherwise identical, would need a unique hostname, ip address and perhaps DNS information. These files take care of the filesystem side of these issues.

Docker can update the resolv.conf when DHCP changes the DNS settings for the host network, the resolv.conf.hash allows Docker to detect changes to the resolv.conf file, avoiding modifications if the user has made changes of their own. The shm directory is used by the shared memory system, each container has its own isolated shared memory by default.

A lot has changed since we started our container. Redisplay the contents of your config.v2.json file:

```
root@ubuntu:/home/user# cat
/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/config.v2.j
```

```
son
{"State":
{"Running":true,"Paused":false,"Restarting":false,"OOMKilled":false,"RemovalInProgress":false,"Dead":fa
lse,"Pid":57538,"StartedAt":"2016-08-22T06:40:28.384874156Z","FinishedAt":"0001-01-
01T00:00:00Z","Health":null},"ID":"ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b","C
reated":"2016-08-22T06:22:05.684955354Z","Managed":false,"Path":"httpd-foreground","Args":[],"Config":
{"Hostname":"ws05","Domainname":"","User":"","AttachStdin":false,"AttachStdout":true,"AttachStderr":tru
e,"ExposedPorts":{"80/tcp":{}},"Tty":false,"OpenStdin":false,"StdinOnce":false,"Env":
["PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin","HTTPD_PREF
IX=/usr/local/apache2","HTTPD_VERSION=2.4.23","HTTPD_SHA1=5101be34ac4a509b245adb70a56690a84fcc4e7f","HT
TPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.23.tar.bz2"],"Cmd":["httpd-
foreground"],"Image":"httpd","Volumes":null,"WorkingDir":"/usr/local/apache2","Entrypoint":null,"OnBuil
d":null,"Labels":
{"copyright":"2016","lic":"apache2"}},"Image":"sha256:462fca6e7913e7758cbc2b476e61b78670bc9ab2573aeb3f6
b22dcd15af32f15","NetworkSettings":
{"Bridge":"","SandboxID":"45235760c6c31b9c6604eae8fccb542da4e10e6c7215317ed22cedf380b3d342","HairpinMod
e":false,"LinkLocalIPv6Address":"","LinkLocalIPv6PrefixLen":0,"Networks":{"bridge":
{"IPAMConfig":null,"Links":null,"Aliases":null,"NetworkID":"1b1a61f6359a1781f4b7e58901508543642b02f3c4e
d2867bf1371972d41d892","EndpointID":"77b3c4aab5d31b5a7e46d1ebf9d7fbb7fe32a82b1e05c690b223dc06e86c6112",
"Gateway":"172.17.0.1","IPAddress":"172.17.0.2","IPPrefixLen":16,"IPv6Gateway":"","GlobalIPv6Address":"
","GlobalIPv6PrefixLen":0,"MacAddress":"02:42:ac:11:00:02"}},"Service":null,"Ports":
{"80/tcp":null},"SandboxKey":"/var/run/docker/netns/45235760c6c3","SecondaryIPAddresses":null,"Secondar
yIPv6Addresses":null,"IsAnonymousEndpoint":false},"LogPath":"/var/lib/docker/containers/ac8a05310b794b1
7a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8a
babb96d2a1b-
json.log","Name":"/websvr5","Driver":"aufs","MountLabel":"","ProcessLabel":"","RestartCount":0,"HasBeen
StartedBefore":false,"HasBeenManuallyStopped":false,"MountPoints":
{},"AppArmorProfile":"","HostnamePath":"/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca905
5d772a6cc842e5b8ababb96d2a1b/hostname","HostsPath":"/var/lib/docker/containers/ac8a05310b794b17a272c6f3
3c8b024ca9055d772a6cc842e5b8ababb96d2a1b/hosts","ShmPath":"/var/lib/docker/containers/ac8a05310b794b17a
272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/shm","ResolvConfPath":"/var/lib/docker/containers/ac8a0
5310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/resolv.conf","SeccompProfile":"","NoNewPriv
ileges":false}
```

Examine the Pid value in the before and after files. As you can see, only running containers have process IDs.

While all of this under the hood exploration is interesting, and certainly informative, it is not very portable. A new version of Docker could move all of these things around, store the metadata in a database or what have you.

In production the best way to collect metadata on a container is to use the inspect command. Run `docker inspect` on your container:

```
root@ubuntu:/home/user# docker inspect websvr5
[
    {
        "Id": "ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b",
        "Created": "2016-08-22T06:22:05.684955354Z",
        "Path": "httpd-foreground",
        "Args": [],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 57538,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2016-08-22T06:40:28.384874156Z",
```

```
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:462fca6e7913e7758cbc2b476e61b78670bc9ab2573aeb3f6b22dcd15af32f15",
        "ResolvConfPath":
"/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/resolv.con
f",
        "HostnamePath":
"/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/hostname",
        "HostsPath":
"/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/hosts",
        "LogPath":
"/var/lib/docker/containers/ac8a05310b794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b/ac8a05310b
794b17a272c6f33c8b024ca9055d772a6cc842e5b8ababb96d2a1b-json.log",
        "Name": "/websvr5",
        "RestartCount": 0,
        "Driver": "aufs",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "",
        "ExecIDs": null,
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
                "Type": "json-file",
                "Config": {}
            },
            "NetworkMode": "default",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
            "AutoRemove": false,
            "VolumeDriver": "",
            "VolumesFrom": null,
            "CapAdd": null,
            "CapDrop": null,
            "Dns": [],
            "DnsOptions": [],
            "DnsSearch": [],
            "ExtraHosts": null,
            "GroupAdd": null,
            "IpcMode": "",
            "Cgroup": "",
            "Links": null,
            "OomScoreAdj": 0,
            "PidMode": "",
            "Privileged": false,
            "PublishAllPorts": false,
            "ReadonlyRootfs": false,
            "SecurityOpt": null,
            "UTSMode": "",
            "UsernsMode": "",
            "ShmSize": 67108864,
            "Runtime": "runc",
            "ConsoleSize": [
                0,
                0
            ],
            "Isolation": "",
```

```
            "CpuShares": 0,
            "Memory": 0,
            "CgroupParent": "",
            "BlkioWeight": 0,
            "BlkioWeightDevice": null,
            "BlkioDeviceReadBps": null,
            "BlkioDeviceWriteBps": null,
            "BlkioDeviceReadIOps": null,
            "BlkioDeviceWriteIOps": null,
            "CpuPeriod": 0,
            "CpuQuota": 0,
            "CpusetCpus": "",
            "CpusetMems": "",
            "Devices": [],
            "DiskQuota": 0,
            "KernelMemory": 0,
            "MemoryReservation": 0,
            "MemorySwap": 0,
            "MemorySwappiness": -1,
            "OomKillDisable": false,
            "PidsLimit": 0,
            "Ulimits": null,
            "CpuCount": 0,
            "CpuPercent": 0,
            "IOMaximumIOps": 0,
            "IOMaximumBandwidth": 0
        },
        "GraphDriver": {
            "Name": "aufs",
            "Data": null
        },
        "Mounts": [],
        "Config": {
            "Hostname": "ws05",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": true,
            "AttachStderr": true,
            "ExposedPorts": {
                "80/tcp": {}
            },
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [

"PATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
                "HTTPD_PREFIX=/usr/local/apache2",
                "HTTPD_VERSION=2.4.23",
                "HTTPD_SHA1=5101be34ac4a509b245adb70a56690a84fcc4e7f",
                "HTTPD_BZ2_URL=https://www.apache.org/dist/httpd/httpd-2.4.23.tar.bz2"
            ],
            "Cmd": [
                "httpd-foreground"
            ],
            "Image": "httpd",
            "Volumes": null,
            "WorkingDir": "/usr/local/apache2",
            "Entrypoint": null,
            "OnBuild": null,
```

```
            "Labels": {
                "copyright": "2016",
                "lic": "apache2"
            }
        },
        "NetworkSettings": {
            "Bridge": "",
            "SandboxID": "45235760c6c31b9c6604eae8fccb542da4e10e6c7215317ed22cedf380b3d342",
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {
                "80/tcp": null
            },
            "SandboxKey": "/var/run/docker/netns/45235760c6c3",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "77b3c4aab5d31b5a7e46d1ebf9d7fbb7fe32a82b1e05c690b223dc06e86c6112",
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "NetworkID": "1b1a61f6359a1781f4b7e58901508543642b02f3c4ed2867bf1371972d41d892",
                    "EndpointID": "77b3c4aab5d31b5a7e46d1ebf9d7fbb7fe32a82b1e05c690b223dc06e86c6112",
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:11:00:02"
                }
            }
        }
    }
]
root@ubuntu:/home/user#
```


Notice that the inspect output has a large section for hostconfig and another for config. Now we know where these come from. You can also see the log file, resolv.conf, hostname, and hosts file references here. We'll look at these and the networking section further later in the course.

Use the inspect filter switch as demonstrated in class to display the following container metadata from your web container:

- Pid
- IPAddress
- Labels
- Environment Variables
- Container name

2. CLI config.json

Using the text as a guide, create a `.docker/config.json` CLI configuration file that uses the psFormat key to customize your default docker ps output. Make your custom format display ID, labels, name, and command at a minimum.

Test and debug your *ps* config.

Note: If you have not issues a docker command requiring Docker to create the config.json it may not exist. You can then create the .docker directory and the config.json yourself.

3. Exploring a containers

- Use `docker inspect` to show the command (Path) used to launch your Docker web server container from earlier in the lab
- Exec a bash shell into the container locate this file (try: # which)
- `cat` the contents of the file
- What does the last line of the file do?
- Open a new terminal and run this command: `$ man exec` , read the first line of the description
- When the httpd-foreground script runs, what is the PID of the startup shell running the script?
- Why is the exec command so important?

4. Signals

In this step we will explore Docker and signals. Run an Ubuntu container that executes the `sleep 30` command:

```
user@ubuntu:~$ docker run -itd ubuntu sleep 30
657d756b5800057c811cab94bbdd3c0365257e21ad6a340374994a05d4405d09
```

This container will sleep for 30 seconds and then exit.

Now run `docker wait` to wait for the Docker container sleeping to exit:

```
user@ubuntu:~$ docker wait 657d; echo "its done"
0
its done
user@ubuntu:~$
```

When the first container exits the `wait` subcommand returns allowing the `echo` command to run. This can be a useful tool in many contexts. Imagine you are creating a script and the commands executing in the script need to be run serially. Containers are typically run asynchronously but you could force the script to wait for the container's completion using the wait command.

You can use the start or restart command to restart the original container after it exits. Try it:

```
user@ubuntu:~$ docker restart 657d
657d
user@ubuntu:~$
```

Imagine you have a flakey service that keeps crashing after running for 5 hours. The team is working on a fix but you need to keep the service running in the mean time. In most cases you would want to run a fresh copy of the container from the image, however if the container can be safely restarted, you can give Docker a restart policy.

Rerun you sleep container with a restart policy of always:

```
user@ubuntu:~$ docker run -itd --restart=always ubuntu sleep 30
000a64a3be582c0a66e7adf0f4a72d5d9cbeae5dcd5b4f647e3ae9533ac7c366

user@ubuntu:~$ docker ps -f Id=000a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
000a64a3be58        ubuntu              "sleep 30"          28 seconds ago      Up 27 seconds
sad_kilby

user@ubuntu:~$ sleep 45

user@ubuntu:~$ docker ps -f Id=000a
CONTAINER ID        IMAGE               COMMAND             CREATED              STATUS
PORTS               NAMES
000a64a3be58        ubuntu              "sleep 30"          About a minute ago   Up 21 seconds
sad_kilby

user@ubuntu:~$
```

In the example above, even when we wait 45 seconds we find the sleep container running. Docker will always restart it if it terminates.

5. Events

Events can help us better understand the happenings within a given Docker engine. For example, in the example above it is difficult to detect the application failure (sleep exiting). We can see the run times are short in the ps listings but there is a better way to track application start and stop events. Run the `docker events` subcommand:

```
user@ubuntu:~$ docker events
2016-08-22T04:47:12.942893269-07:00 container die
000a64a3be582c0a66e7adf0f4a72d5d9cbeae5dcd5b4f647e3ae9533ac7c366 (exitCode=0, image=ubuntu,
name=sad_kilby)
2016-08-22T04:47:13.408367724-07:00 container start
000a64a3be582c0a66e7adf0f4a72d5d9cbeae5dcd5b4f647e3ae9533ac7c366 (image=ubuntu, name=sad_kilby)
2016-08-22T04:47:43.529352180-07:00 container die
000a64a3be582c0a66e7adf0f4a72d5d9cbeae5dcd5b4f647e3ae9533ac7c366 (exitCode=0, image=ubuntu,
name=sad_kilby)
2016-08-22T04:47:44.049379160-07:00 container start
000a64a3be582c0a66e7adf0f4a72d5d9cbeae5dcd5b4f647e3ae9533ac7c366 (image=ubuntu, name=sad_kilby)
```

Within 30 seconds you should see your sleep container exit (when the sleep expires) and then get started by Docker due to the restart policy.

6. Cleanup

After you have finished exploring, stop and remove all of the containers you started/created in this lab. Remember that you can refer to containers by name or ID.

Congratulations you have completed the advanced Controlling Containers lab!