Cloud Foundry Application Runtime

Lab 13 – Running Docker Containers on CFAR

The Cloud Foundry Garden container engine supports OCI container images and can pull images from any system supporting the docker registry V2 API specification. In this lab we'll try running a docker container on cloud foundry and then exploring the container.

1. Run a public image on CFAR

To deploy a Docker image from a Docker Hub repository, run cf push APP-NAME --docker-image REPO/IMAGE:TAG. Replace the placeholder values in the command as follows:

- APP-NAME: The name of the app being pushed
- REPO: The name of the repository where the image is stored
- IMAGE: The name of an image from Docker Hub
- TAG: (Optional) The tag or version for the image

Push a standard nginx container from docker hub to your cloud foundry space (give the app a unique name):

```
user@ubuntu:~$ cf push my-app --docker-image nginx
Pushing app rxm-app33 to org rx-m.com / space development as randy.abernethy@rx-m.com...
Getting app info...
Creating app with these attributes...
+ name:           rxm-app33
+ docker image:   nginx
  routes:
+   rxm-app33.cfapps.io

Creating app rxm-app33...
Mapping routes...

Staging app and tracing logs...
   Cell a9e19260-c172-41da-a126-36e77b5ed4a8 successfully created container for instance 7f2f3118-0ada-4490-a541-
e24d887be803
   Staging...
   Staging process started ...
   Staging process finished
   Exit status 0
   Staging Complete
   Cell a9e19260-c172-41da-a126-36e77b5ed4a8 stopping instance 7f2f3118-0ada-4490-a541-e24d887be803
   Cell a9e19260-c172-41da-a126-36e77b5ed4a8 destroying container for instance 7f2f3118-0ada-4490-a541-e24d887be803
   Cell a9e19260-c172-41da-a126-36e77b5ed4a8 successfully destroyed container for instance 7f2f3118-0ada-4490-a541-
e24d887be803

Waiting for app to start...

name:              rxm-app33
requested state:   started
instances:         1/1
usage:             1G x 1 instances
routes:            rxm-app33.cfapps.io
last uploaded:     Fri 29 Jun 10:34:26 PDT 2018
stack:             cflinuxfs2
docker image:      nginx
start command:     nginx -g daemon off;

     state     since                  cpu    memory    disk      details
#0   running   2018-06-29T17:34:36Z   0.0%   0 of 1G   0 of 1G

user@ubuntu:~$
```

Many Docker registries control access to images by authenticating with a username and password. You can add user and password authentication information to cf push if required:

- The CF_DOCKER_PASSWORD environment variable defines the password
- The --docker-username command line argument defines the username

2. Exploring the container

Let's use the Cloud Foundry ssh command to explore the contents of our new running container:

```
user@ubuntu:~$ cf ssh rxm-app33
```

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

The ssh command uses ssh to connect to the container. This can be helpful when debugging, inspection and monitoring processes.

List the files inside the container:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# ls -l /
total 8
drwxr-xr-x   2 root   root    4096 Jun 29 18:03 bin
drwxr-xr-x   2 root   root       6 Feb 23 23:23 boot
drwxr-xr-x   4 root   root     179 Jun 29 18:03 dev
drwxr-xr-x   1 root   root      97 Jun 29 18:03 etc
drwxr-xr-x   2 root   root       6 Feb 23 23:23 home
drwxr-xr-x   1 root   root      45 Jun 29 18:03 lib
drwxr-xr-x   2 root   root      34 Jun 29 18:03 lib64
drwxr-xr-x   2 root   root       6 Jun 25 00:00 media
drwxr-xr-x   2 root   root       6 Jun 25 00:00 mnt
drwxr-xr-x   2 root   root       6 Jun 25 00:00 opt
dr-xr-xr-x 871 nobody nogroup    0 Jun 29 18:03 proc
drwx------   2 root   root      37 Jun 29 18:03 root
drwxr-xr-x   1 root   root      23 Jun 29 18:03 run
drwxr-xr-x   2 root   root    4096 Jun 29 18:03 sbin
drwxr-xr-x   2 root   root       6 Jun 25 00:00 srv
dr-xr-xr-x  13 nobody nogroup    0 Jun 23 08:02 sys
drwxrwxrwt   1 root   root      42 Jun 29 18:03 tmp
drwxr-xr-x   1 root   root      66 Jun 29 18:03 usr
drwxr-xr-x   1 root   root      19 Jun 29 18:03 var
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

The file system does not have a kernel! This is because it doesn't need one, application containers always use the kernel of the underlying host system.

Try listing all of the running processes:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# ps -ef
bash: ps: command not found
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

This container rootfs does not include a ps binary so we can not run the ps command (!). Instead try listing the files : in the standard Unix proc directory:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# ls -l /proc

total 0
dr-xr-xr-x  9 root   root       0 Jun 29 18:10 1
dr-xr-xr-x  9 root   root       0 Jun 29 18:10 102
dr-xr-xr-x  9 root   root       0 Jun 29 18:10 111
dr-xr-xr-x  9 root   root       0 Jun 29 18:10 35
dr-xr-xr-x  9 nginx  nginx      0 Jun 29 18:10 50
dr-xr-xr-x  9 root   root       0 Jun 29 18:03 7
dr-xr-xr-x  9 root   root       0 Jun 29 18:10 71
dr-xr-xr-x  9 root   root       0 Jun 29 18:03 8
dr-xr-xr-x  9 root   root       0 Jun 29 18:03 89
dr-xr-xr-x  2 nobody nogroup    0 Jun 29 18:10 acpi
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 buddyinfo
dr-xr-xr-x  4 nobody nogroup    0 Jun 29 18:10 bus
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 cgroups
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 cmdline
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 consoles
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 cpuinfo
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 crypto
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 devices
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 diskstats
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 dma
dr-xr-xr-x  2 nobody nogroup    0 Jun 29 18:10 driver
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 execdomains
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 fb
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 filesystems
dr-xr-xr-x 10 nobody nogroup    0 Jun 29 18:10 fs
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 interrupts
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 iomem
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 ioports
dr-xr-xr-x 50 nobody nogroup    0 Jun 29 18:10 irq
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 kallsyms
```

```
crw-rw-rw-  1 nobody nogroup 1, 3 Jun 23 08:01 kcore
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 key-users
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 keys
-r--------  1 nobody nogroup    0 Jun 29 18:10 kmsg
-r--------  1 nobody nogroup    0 Jun 29 18:10 kpagecgroup
-r--------  1 nobody nogroup    0 Jun 29 18:10 kpagecount
-r--------  1 nobody nogroup    0 Jun 29 18:10 kpageflags
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 loadavg
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 locks
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 mdstat
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 meminfo
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 misc
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 modules
lrwxrwxrwx  1 nobody nogroup   11 Jun 29 18:10 mounts -> self/mounts
-rw-r--r--  1 nobody nogroup    0 Jun 29 18:10 mtrr
lrwxrwxrwx  1 nobody nogroup    8 Jun 29 18:10 net -> self/net
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 pagetypeinfo
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 partitions
crw-rw-rw-  1 nobody nogroup 1, 3 Jun 23 08:01 sched_debug
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 schedstat
drwxrwxrwt  2 root   root      40 Jun 29 18:03 scsi
lrwxrwxrwx  1 nobody nogroup    0 Jun 29 18:03 self -> 111
-r--------  1 nobody nogroup    0 Jun 29 18:10 slabinfo
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 softirqs
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 stat
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 swaps
dr-xr-xr-x  1 nobody nogroup    0 Jun 29 18:03 sys
--w-------  1 nobody nogroup    0 Jun 29 18:10 sysrq-trigger
dr-xr-xr-x  2 nobody nogroup    0 Jun 29 18:10 sysvipc
lrwxrwxrwx  1 nobody nogroup    0 Jun 29 18:03 thread-self -> 111/task/111
crw-rw-rw-  1 nobody nogroup 1, 3 Jun 23 08:01 timer_list
crw-rw-rw-  1 nobody nogroup 1, 3 Jun 23 08:01 timer_stats
dr-xr-xr-x  4 nobody nogroup    0 Jun 29 18:10 tty
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 uptime
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 version
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 version_signature
-r--------  1 nobody nogroup    0 Jun 29 18:10 vmallocinfo
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 vmstat
dr-xr-xr-x  2 nobody nogroup    0 Jun 29 18:10 xen
-r--r--r--  1 nobody nogroup    0 Jun 29 18:10 zoneinfo

root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

The /proc filesystem is used on Unix systems to give privileged users "file system" style access to system information. Note that most of the files have a size of 0. This is because they are not really files, they are simply filesystem constructs allowing us to query attributed of the system.

All of the files with just a number for a name are actually process IDs for the running processes. List them:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# ls -l /proc/ | grep ' [0-9]*$'

total 0
dr-xr-xr-x  9 root   root       0 Jun 29 18:18 1
dr-xr-xr-x  9 root   root       0 Jun 29 18:18 102
dr-xr-xr-x  9 root   root       0 Jun 29 18:18 135
dr-xr-xr-x  9 root   root       0 Jun 29 18:18 136
dr-xr-xr-x  9 root   root       0 Jun 29 18:18 35
dr-xr-xr-x  9 nginx  nginx      0 Jun 29 18:18 50
dr-xr-xr-x  9 root   root       0 Jun 29 18:03 7
dr-xr-xr-x  9 root   root       0 Jun 29 18:18 71
dr-xr-xr-x  9 root   root       0 Jun 29 18:03 8
dr-xr-xr-x  9 root   root       0 Jun 29 18:03 89
lrwxrwxrwx  1 nobody nogroup    0 Jun 29 18:03 self -> 135

root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

This command shows all of the files with a number for a name. The self link always points to the currently running process (which would have been the ls command in this example).

In Unix systems, process 1 is the first process executed after system boot or, in the case of containers with a process namespace, the first process to run in the container. List the contents of the "1" directory:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# ls -l /proc/1

total 0
dr-xr-xr-x 2 root root 0 Jun 29 18:20 attr
```

```
-rw-r--r-- 1 root root 0 Jun 29 18:20 autogroup
-r-------- 1 root root 0 Jun 29 18:20 auxv
-r--r--r-- 1 root root 0 Jun 29 18:20 cgroup
--w------- 1 root root 0 Jun 29 18:20 clear_refs
-r--r--r-- 1 root root 0 Jun 29 18:20 cmdline
-rw-r--r-- 1 root root 0 Jun 29 18:20 comm
-rw-r--r-- 1 root root 0 Jun 29 18:20 coredump_filter
-r--r--r-- 1 root root 0 Jun 29 18:20 cpuset
lrwxrwxrwx 1 root root 0 Jun 29 18:20 cwd -> /
-r-------- 1 root root 0 Jun 29 18:20 environ
lrwxrwxrwx 1 root root 0 Jun 29 18:20 exe -> /tmp/garden-init
dr-x------ 2 root root 0 Jun 29 18:20 fd
dr-x------ 2 root root 0 Jun 29 18:20 fdinfo
-rw-r--r-- 1 root root 0 Jun 29 18:20 gid_map
-r-------- 1 root root 0 Jun 29 18:20 io
-r--r--r-- 1 root root 0 Jun 29 18:20 limits
-rw-r--r-- 1 root root 0 Jun 29 18:20 loginuid
dr-x------ 2 root root 0 Jun 29 18:20 map_files
-r--r--r-- 1 root root 0 Jun 29 18:20 maps
-rw------- 1 root root 0 Jun 29 18:20 mem
-r--r--r-- 1 root root 0 Jun 29 18:20 mountinfo
-r--r--r-- 1 root root 0 Jun 29 18:20 mounts
-r-------- 1 root root 0 Jun 29 18:20 mountstats
dr-xr-xr-x 6 root root 0 Jun 29 18:20 net
dr-x--x--x 2 root root 0 Jun 29 18:20 ns
-r--r--r-- 1 root root 0 Jun 29 18:20 numa_maps
-rw-r--r-- 1 root root 0 Jun 29 18:20 oom_adj
-r--r--r-- 1 root root 0 Jun 29 18:20 oom_score
-rw-r--r-- 1 root root 0 Jun 29 18:20 oom_score_adj
-r-------- 1 root root 0 Jun 29 18:20 pagemap
-r-------- 1 root root 0 Jun 29 18:20 personality
-rw-r--r-- 1 root root 0 Jun 29 18:20 projid_map
lrwxrwxrwx 1 root root 0 Jun 29 18:20 root -> /
-rw-r--r-- 1 root root 0 Jun 29 18:20 sched
-r--r--r-- 1 root root 0 Jun 29 18:20 schedstat
-r--r--r-- 1 root root 0 Jun 29 18:20 sessionid
-rw-r--r-- 1 root root 0 Jun 29 18:20 setgroups
-r--r--r-- 1 root root 0 Jun 29 18:20 smaps
-r-------- 1 root root 0 Jun 29 18:20 stack
-r--r--r-- 1 root root 0 Jun 29 18:20 stat
-r--r--r-- 1 root root 0 Jun 29 18:20 statm
-r--r--r-- 1 root root 0 Jun 29 18:20 status
-r-------- 1 root root 0 Jun 29 18:20 syscall
dr-xr-xr-x 3 root root 0 Jun 29 18:20 task
-r--r--r-- 1 root root 0 Jun 29 18:20 timers
-rw-r--r-- 1 root root 0 Jun 29 18:20 uid_map
-r--r--r-- 1 root root 0 Jun 29 18:20 wchan

root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

Each of these files describes some attribute of the PID 1 process. To display the command line the process is running cat the contents of the cmdline file:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# cat /proc/1/cmdline

/tmp/garden-init

root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

The CF Garden container manager launches all containers with an initialization process called garden-init. Unix systems create processes with sequential process ids. Some processes exit right away leaving gaps in the id sequence. List the command line of the next sequential process id after 1:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# cat /proc/7/cmdline

/tmp/lifecycle/diego-sshd--allowedKeyExchanges=--address=0.0.0.0:2222--allowUnauthenticatedClients=false--
inheritDaemonEnv=true--allowedCiphers=--allowedMACs=--logLevel=fatal--debugAddr=

root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

This is the Diego sshd daemon that we are connected to. List the next process:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# cat /proc/8/cmdline
```

```
nginx: master process /usr/sbin/nginx -g daemon off;

root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~#
```

This is the actual container image process we intended to run.

Explore some more if you like, then exit the ssh shell:

```
root@2f3f1dfc-e9ec-4f2a-4d1e-085e:~# exit

exit

user@ubuntu:~$
```

CHALLANGE (optional) Run a private image

In this challange step you will push your trash can inventory image from MSA lab 3 to Docker Hub and then use sf push to deploy it to cloud foundry.

- Navigate to https://hub.docker.com using any browser.
- Create a free account (or login to your existing account)
- Push you trash can inventory image from MSA lab 3 to your private dockerhub account
  - Remember you can create names for images using "docker image tag", e.g. for an account named fredsmith and an image inv you would need to tag the image with a name like "fredsmith/inv" to push it to the fredsmith account on docker hub
- Now use "cf push" as per above to run your container in your PWS space
  - Remember you will need to add your name and password to the cf push command

Congratulations, you have completed the Lab.