Docker On i.MX6UL With ARM Ubuntu

CAS Team

Sep 25, 2019

Environment

HW: i.MX6UL EVK
SW: L4.14.98 GA

Target: Install Ubuntu xenial(16.04) + Docker

STEPS

- Create Basic Ubuntu rootfs
- Install Docker
- Modified the Kernel Configuration
- Create Docker Demo SDCard Image
- Test Docker

CREATE BASIC UBUNTU ROOTFS

Host preparation

- To Install xenial(16.04), please make sure the host ubuntu OS version is not lower than xenial(16.04)
- Install the necessary software

  ```
  sudo apt-get install qemu-user-static debootstrap binfmt-support
  ```
- workspace

  ```
  mkdir ~/workspace
  mkdir -p ~/workspace/mnt      # For mount
  ```
  prepare L4.14.98 Linux source code in the workspace
  prepare the L4.14.98 GA Linux Binary Demo image in the workspace

```
workspace/
|-- fsl-image-validation-imx-imx6ul7d.sdcard --- L4.14.98 GA Linux Binary Demo image
|-- linux-imx   ---  L4.14.98 Linux source code
`-- mnt
```

Debootstrap to create ARM32(armhf) rootfs

```
distro=xenial
arch=armhf
target=rootfs_${distro}_${arch}
mkdir ${target}

sudo debootstrap --arch=${arch} --foreign ${distro} ${target}

# copy qemu-arm-static binary and
# resolv.conf from host to target
sudo cp /usr/bin/qemu-arm-static ${target}/usr/bin
sudo cp /etc/resolv.conf  ${target}/etc/
#sudo chroot rootfs_xenial_armhf
sudo chroot ${target}

# now we are in the chroot, run below:
distro=xenial
arch=armhf

export LANG=en_US.UTF-8
export LC_ALL=C.UTF-8
```

```
# setup second stage
/debootstrap/debootstrap --second-stage
```

Now we have very basic ubuntu rootfs

#Optional but suggest

```
apt-get install -y openssh-server vim ntpdate

exit
```

```
workspace/
|-- fsl-image-validation-imx-imx6ul7d.sdcard
|-- linux-imx
|-- mnt
`-- rootfs_xenial_armhf
```

Modify the ARM32(armhf) rootfs

The following can be done in “sudo chroot ${target}”
or directly from host side but need **sudo like “sudo vim”**

edit ${target}/etc/apt/sources.list and add below:

```
deb http://ports.ubuntu.com/ubuntu-ports xenial main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports xenial-updates main restricted universe multiverse
```

edit ${target}/etc/fstab and add below:

```
/dev/root          /        auto      defaults      1  1
```

edit ${target}/etc/hostname

```
xenial-armhf                         #which is ${distro}-${arch}
```

edit ${target}/etc/hosts and add below:

```
127.0.0.1 localhost xenial-armhf
```

Note: xenial-armhf **is from /etc/hostname**

edit ${target}/etc/network/interfaces and add below:

```
source-directory /etc/network/interfaces.d
iface eth0 inet dhcp
auto eth0
```

Modify the ARM32(armhf) rootfs (Cont.)

Make sure we are in “sudo chroot ${target}” to do below:

```
sudo chroot ${target}
export LANG=en_US.UTF-8
export LC_ALL=C.UTF-8

useradd user -g sudo -m
# add to tty group for tty access
usermod -a -G tty user
# add to dialout group for UART access
usermod -a -G dialout user
# add to sudo group for root access
usermod -a -G sudo user
# Set root password
passwd
# Set user password
passwd user

# Followings are optional
locale-gen en_US.UTF-8
localectl set-locale LANG=en_US.UTF-8
localectl set-locale LC_ALL=C.UTF-8
```

INSTALL DOCKER

Reference Document

https://docs.docker.com/install/linux/docker-ce/ubuntu/

Docker Installation

Make sure we are still in “sudo chroot ${target}” to do below:

```
sudo chroot ${target}
export LANG=en_US.UTF-8
export LC_ALL=C.UTF-8

apt-get update
apt-get install -y libltdl7 libseccomp2
apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
apt-key fingerprint 0EBFCD88

add-apt-repository  "deb [arch=armhf] https://download.docker.com/linux/ubuntu  $(lsb_release -cs)  stable"
apt-get update


apt-get install -y docker-ce docker-ce-cli containerd.io

exit
```

After ARM side installation is finished, ${target}/usr/bin/qemu-arm-static can be deleted.

Post-installation(optional)

Make sure we are still in “sudo chroot ${target}”

```
groupadd docker
usermod -a -G docker user

apt-get clean
```

After ARM side installation is finished, ${target}/usr/bin/qemu-arm-static can be deleted.

MODIFIED THE KERNEL CONFIGURATION

Docker Linux Kernel Configuration Generally Necessary

CONFIG_NAMESPACES
CONFIG_NET_NS
CONFIG_PID_NS
CONFIG_IPC_NS
CONFIG_UTS_NS
CONFIG_CGROUPS
CONFIG_CGROUP_CPUACCT
CONFIG_CGROUP_DEVICE
CONFIG_CGROUP_FREEZER
CONFIG_CGROUP_SCHED
CONFIG_CPUSETS
CONFIG_MEMCG
CONFIG_KEYS
CONFIG_VETH

CONFIG_BRIDGE
CONFIG_BRIDGE_NETFILTER
CONFIG_NF_NAT_IPV4
CONFIG_IP_NF_FILTER
CONFIG_IP_NF_TARGET_MASQUERADE
CONFIG_NETFILTER_XT_MATCH_ADDRTYPE
CONFIG_NETFILTER_XT_MATCH_CONNTRACK
CONFIG_NETFILTER_XT_MATCH_IPVS
CONFIG_IP_NF_NAT
CONFIG_NF_NAT
CONFIG_NF_NAT_NEEDED
CONFIG_POSIX_MQUEUE

Docker Linux Kernel Configuration Optional Features

CONFIG_USER_NS
CONFIG_SECCOMP
CONFIG_CGROUP_PIDS
CONFIG_MEMCG_SWAP
CONFIG_MEMCG_SWAP_ENABLED boot option "swapaccount=1"
CONFIG_LEGACY_VSYSCALL_EMULATE
CONFIG_MEMCG_KMEM
CONFIG_BLK_CGROUP
CONFIG_BLK_DEV_THROTTLING
CONFIG_IOSCHED_CFQ
CONFIG_CFQ_GROUP_IOSCHED
CONFIG_CGROUP_PERF
CONFIG_CGROUP_HUGETLB

CONFIG_CGROUP_HUGETLB
CONFIG_NET_CLS_CGROUP
CONFIG_CGROUP_NET_PRIO
CONFIG_CFS_BANDWIDTH
CONFIG_FAIR_GROUP_SCHED
CONFIG_RT_GROUP_SCHED
CONFIG_IP_NF_TARGET_REDIRECT
CONFIG_IP_VS
CONFIG_IP_VS_NFCT
CONFIG_IP_VS_PROTO_TCP
CONFIG_IP_VS_PROTO_UDP
CONFIG_IP_VS_RR
CONFIG_EXT4_FS
CONFIG_EXT4_FS_POSIX_ACL
CONFIG_EXT4_FS_SECURITY

Docker Linux Kernel Configuration Network Drivers

```
"overlay":
   CONFIG_VXLAN
   CONFIG_BRIDGE_VLAN_FILTERING
   Optional (for encrypted networks):
   CONFIG_CRYPTO
   CONFIG_CRYPTO_AEAD
   CONFIG_CRYPTO_GCM
   CONFIG_CRYPTO_SEQIV
   CONFIG_CRYPTO_GHASH
   CONFIG_XFRM XFRM_USER
   CONFIG_XFRM_ALGO
   CONFIG_INET_ESP
   CONFIG_INET_XFRM_MODE_TRANSPORT
```

```
"ipvlan":
  CONFIG_IPVLAN
"macvlan":
  CONFIG_MACVLAN
  CONFIG_DUMMY
"ftp,tftp client in container":
  CONFIG_NF_NAT_FTP
  CONFIG_NF_CONNTRACK_FTP
  CONFIG_NF_NAT_TFTP
  CONFIG_NF_CONNTRACK_TFTP
```

Docker Linux Kernel Configuration Storage Drivers

```
"aufs":
  CONFIG_AUFS_FS
"btrfs":
  CONFIG_BTRFS_FS
  CONFIG_BTRFS_FS_POSIX_ACL
"devicemapper":
  CONFIG_BLK_DEV_DM
  CONFIG_DM_THIN_PROVISIONING
"overlay":
  CONFIG_OVERLAY_FS
```

Modified the Kernel configuration

Just for reference:
During kernel reconfiguration process, it's possible that your kernel is still missing modules that are required for docker to function properly; you can try running below script to see what's missing:
https://github.com/docker/docker/blob/master/contrib/check-config.sh

After modification of the kernel configuration, user can use check_config.sh to check your kernel configuration file, see if there is missing on the **general necessary** options.

```
chmod +x check-config.sh
dos2unix check-config.sh

 source /opt/fsl-imx-fb/4.14-sumo/environment-setup-cortexa7hf-neon-poky-linux-gnueabi

make imx_v7_defconfig -C linux-imx
./check-config.sh  linux-imx/.config
```

Please ignore CONFIG_DEVPTS_MULTIPLE_INSTANCES Missing

```
workspace/
|-- check-config.sh
|-- fsl-image-validation-imx-imx6ul7d.sdcard
|-- linux-imx
|-- mnt
`-- rootfs_xenial_armhf
```

Modified the Kernel configuration(Cont.)

L4.14.98 GA, Need to especially enable:

CONFIG_NAMESPACES, CONFIG_CGROUPS,
CONFIG_CGROUP_**,
CONFIG_BRIDGE,
CONFIG_BRIDGE_NETFILTER,
CONFIG_VETH,
CONFIG_IP_NF_IPTABLES,
CONFIG_BRIDGE_NF_EBTABLES,
CONFIG_XFRM_USER,
CONFIG_NF_CT_NETLINK…
CONFIG_OVERLAY_FS, CONFIG_MACVLAN, CONFIG_BTRFS_FS, CONFIG_BTRFS_FS_POSIX_AL

```
make imx_v7_defconfig -C linux-imx
make menuconfig -C linux-imx
```

Note: After modification of the kernel configuration. Can use the make savedefconfig to generate the new default configuration linux-imx/defconfig and copy to arch/arm/configs/imx_v7_docker_defconfig

```
make savedefconfig -C linux-imx
cp linux-imx/defconfig linux-imx/arch/arm/configs/imx_v7_docker_defconfig
```

GENERATE KERNEL/MODULES AND INSTALL KERNEL/MODULES

Generate Kernel/modules and modules

```
distro=xenial
arch=armhf
target=rootfs_${distro}_${arch}
```

```
workspace/
|-- fsl-image-validation-imx-imx6ul7d.sdcard
|-- linux-imx
|-- mnt
`-- rootfs_xenial_armhf
```

```
source /opt/fsl-imx-fb/4.14-sumo/environment-setup-cortexa7hf-neon-poky-linux-gnueabi
```

```
make imx_v7_docker_defconfig -C linux-imx
LDFLAGS=""  CC="$CC"  make -j8 zImage modules -C linux-imx
```

Now, we have new kernel Image and modules

```
sudo  make modules_install INSTALL_MOD_PATH=$(pwd)/${target} ARCH=arm  LDFLAGS="" -C linux-imx
```

INSTALL_MOD_PATH needs FULL path

Note: distro=xenial
arch=armhf

${target} → target=rootfs_${distro}_${arch}
→ rootf_xenial_armhf

CREATE DOCKER DEMO SDCARD IMAGE

resize SDCard rootfs partition

```
truncate -s 7G fsl-image-validation-imx-imx6ul7d.sdcard
sudo parted fsl-image-validation-imx-imx6ul7d.sdcard  unit MiB print
Model:  (file)
Disk fsl-image-validation-imx-imx6ul7d.sdcard: 7168MiB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start    End      Size     Type     File system  Flags
 1      4.00MiB  36.0MiB  32.0MiB  primary  fat16        lba
 2      36.0MiB  980MiB   944MiB   primary  ext4
sudo parted fsl-image-validation-imx-imx6ul7d.sdcard resizepart 2 7160MiB
sudo parted fsl-image-validation-imx-imx6ul7d.sdcard unit MiB print
Model:  (file)
fsl-image-validation-imx-imx6ul7d.sdcard: 7168MiB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start    End      Size     Type     File system  Flags
 1      4.00MiB  36.0MiB  32.0MiB  primary  fat16        lba
 2      36.0MiB  7160MiB  7124MiB  primary  ext4

sudo kpartx -av fsl-image-validation-imx-imx6ul7d.sdcard
sudo e2fsck  -f /dev/mapper/loop0p2
sudo resize2fs /dev/mapper/loop0p2
sudo kpartx -d  fsl-image-validation-imx-imx6ul7d.sdcard
```

Now the rootfs is resized to 7160M.   You can also set as other values like 2G.

Note: The operations could be done on a “real” sdcard.

Replace with Ubuntu Rootfs and update Linux Kernel Image

```
sudo kpartx -av fsl-image-validation-imx-imx6ul7d.sdcard

sudo mkfs.ext4 /dev/mapper/loop0p2
sudo mount /dev/mapper/loop0p2 mnt/
sudo cp -rf  ${target}/*  mnt/
sudo umount mnt

sudo mount /dev/mapper/loop0p1 mnt/
sudo cp  -rf linux-imx/arch/arm/boot/zImage mnt/
sudo umount mnt

sudo kpartx -d  fsl-image-validation-imx-imx6ul7d.sdcard
```

Done!
Use Linux dd command or windows win32diskimager to burn to SDCard for test.

CREATE DOCKER DEMO SDCARD IMAGE

Test Docker

Boot the board and connect the ethernet and make sure the board can access internet:

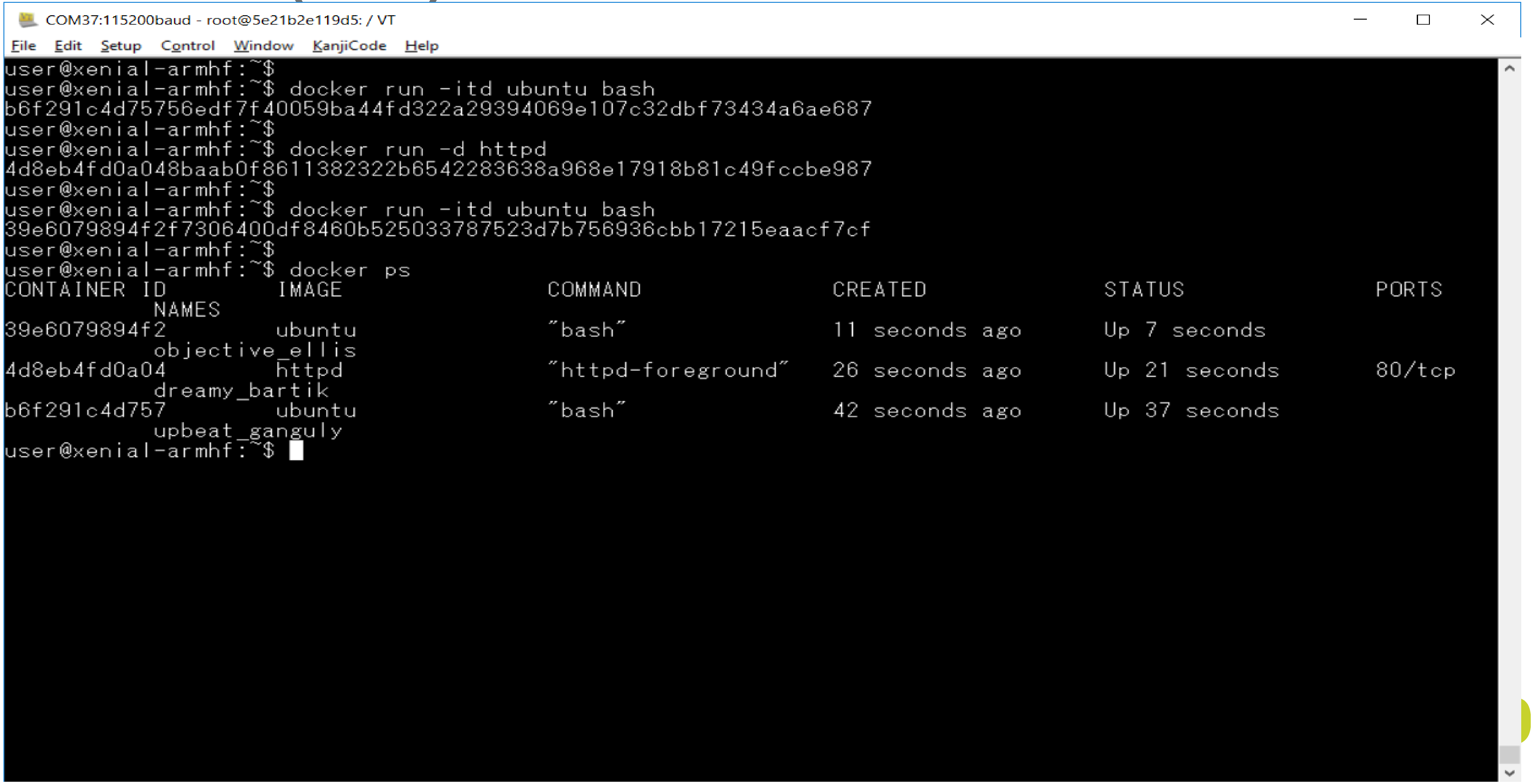
```
dhclient eth0
#ping baidu.com
date +%Y%m%d -s “20190917”    //set the date to current real date
docker version
docker pull hello-world
docker run hello-world

docker run -it ubuntu bash    //may need to resize the sdcard to support bigger space, refer to later slides.
```

Just for debug:

```
systemctl status docker
docker ps -a
systemctl stop docker
sudo dockerd   //see the error info
```

Test Docker(Cont.)


COM37:115200baud - root@5e21b2e119d5: / VT

File Edit Setup Control Window KanjiCode Help

```
user@xenial-armhf:~$
user@xenial-armhf:~$ docker run -itd ubuntu bash
b6f291c4d75756edf7f40059ba44fd322a29394069e107c32dbf73434a6ae687
user@xenial-armhf:~$
user@xenial-armhf:~$ docker run -d httpd
4d8eb4fd0a048baab0f8611382322b6542283638a968e17918b81c49fccbe987
user@xenial-armhf:~$
user@xenial-armhf:~$ docker run -itd ubuntu bash
39e6079894f2f7306400df8460b525033787523d7b756936cbb17215eaacf7cf
user@xenial-armhf:~$
user@xenial-armhf:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
           NAMES
39e6079894f2        ubuntu              "bash"                   11 seconds ago      Up 7 seconds
           objective_ellis
4d8eb4fd0a04        httpd               "httpd-foreground"       26 seconds ago      Up 21 seconds       80/tcp
           dreamy_bartik
b6f291c4d757        ubuntu              "bash"                   42 seconds ago      Up 37 seconds
           upbeat_ganguly
user@xenial-armhf:~$
```

SECURE CONNECTIONS
FOR A SMARTER WORLD