## GNU/Linux Administration - Support #683

# Linux Containers on Arch Linux

10/22/2015 10:57 AM - Daniel Curtis

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 10/22/2015 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Daniel Curtis | **% Done:** | 100% |
| **Category:** | Jails / Container | **Estimated time:** | 3.00 hours |
| **Target version:** | Arch Linux | **Spent time:** | 7.00 hours |

**Description**

This is a simple guide for setting up and using Linux Containers on Arch Linux.

## Prepare the Environment

- Make sure the system is up to date:

```
pacman -Syu
```

- Install yay

## Set Up The Network

- Install netctl:

```
pacman -S netctl
```

### (config 1) Bridged Wired Connection

Bridge Internet-shared - This example will bridge network interface eth0 and configure a static IP for the bridge.

- Create the netctl profile:

```
nano /etc/netctl/lxcbridge
```

  - And add/modify the following:

```
Description="LXC Bridge"
Interface=br0
Connection=bridge
BindsToInterfaces=(eth0)
IP=dhcp
SkipForwardingDelay=yes
```

- After changes are made, make sure to re-enable and restart the bridge:

```
netctl enable lxcbridge
netctl start lxcbridge
```

- Enable IP Forwarding persist at boot:

```
echo 'net.ipv4.ip_forward=1' >> /etc/sysctl.d/40-ip-forward.conf
```

- And also apply this iptables rule:

  ```
  iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
  ```

  - To make changes persist upon reboot:

    ```
    iptables-save > /etc/iptables/iptables.rules
    systemctl enable iptables
    systemctl start iptables
    ```

- Edit the default lxc config:
  nano /etc/lxc/default.conf
  - Add add the following to set all newly created containers to use the br0 interface:

    ```
    lxc.net.0.type=veth
    lxc.net.0.link=br0
    lxc.net.0.flags=up
    ```

## (config 2) NAT Wireless Connection

NAT Internet-shared - This example will setup the network interface wlan0 and configure a static IP for the bridge.

- Install dnsmasq:

  ```
  pacman -S dnsmasq
  ```

- Make sure that forwarding is turned on to support NAT:

  ```
  echo 1 > /proc/sys/net/ipv4/ip_forward
  ```

- Make the previous change persistent after a reboot:

  ```
  echo 'net.ipv4.ip_forward=1' >> /etc/sysctl.d/30-ipforward.conf
  echo 'net.ipv6.conf.default.forwarding=1' >> /etc/sysctl.d/30-ipforward.conf
  echo 'net.ipv6.conf.all.forwarding=1' >> /etc/sysctl.d/30-ipforward.conf
  ```

- Create the netctl profile:

  ```
  nano /etc/netctl/lxcnatbridge
  ```

  - And add/modify the following:

    ```
    Description="LXC NAT Bridge"
    Interface=natbr0
    Connection=bridge
    IP=static
    Address=192.168.10.200/24
    DNS=192.168.1.1
    SkipForwardingDelay=yes
    ExecUpPost="/usr/local/bin/natbr0-up"
    ```

- Create the nat script:

```
nano /usr/local/bin/natbr0-up
```

- And add the following:

  ```
  #!/bin/sh
  # Script to setup NAT iptables masquerade rules; and dnsmasq for DHCP and DNS.

  # This is the address assigned to the bridge at boot
  BRADDR=192.168.10.200

  # DHCP IP address range for containers
  BRRANGE=192.168.10.201,192.168.10.250

  # Configure iptables rules for NAT
  iptables -A FORWARD -i natbr0 -s ${BRADDR}/24 -m conntrack --ctstate NEW -j ACCEPT
  iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
  iptables -A POSTROUTING -t nat -j MASQUERADE

  # (Optional) Forward port 2222 to host behind NAT bridge
  # iptables -t nat -A PREROUTING -p tcp --dport 2222 -j DNAT --to 192.168.10.200:22

  # Start dnsmasq DNS/DHCP server for the network attached to the NAT interface.
  dnsmasq --bind-interfaces --conf-file= --listen-address $BRADDR --except-interface lo --dh
  cp-range $BRRANGE --dhcp-lease-max=253 --dhcp-no-override
  ```

- Make sure it is executable by doing

```
chmod +x /usr/local/bin/natbr0-up
```

- Start and enable the NAT bridge at boot:

```
netctl enable lxcnatbridge
netctl start lxcnatbridge
```

- Edit the default lxc config:
  nano /etc/lxc/default.conf
  - Add add the following to set all newly created containers to use the natbr0 interface:

  ```
  lxc.net.0.type=veth
  lxc.net.0.link=natbr0
  lxc.net.0.flags=up
  ```

## Install LXC

- Install lxc, bridge-utils, and arch-install-scripts:

```
pacman -S lxc bridge-utils arch-install-scripts
```

  - (*Optional*) Install extra packages for lxc:

  ```
  pacman -S lua lua-filesystem lua-alt-getopt
  ```

- Test that the system is correctly configured

```
lxc-checkconfig
```

- The output should be similar to:

  ```
  --- Namespaces ---
  Namespaces: enabled
  Utsname namespace: enabled
  Ipc namespace: enabled
  Pid namespace: enabled
  User namespace: missing
  Network namespace: enabled
  Multiple /dev/pts instances: enabled

  --- Control groups ---
  Cgroup: enabled
  Cgroup clone_children flag: enabled
  Cgroup device: enabled
  Cgroup sched: enabled
  Cgroup cpu account: enabled
  Cgroup memory controller: enabled
  Cgroup cpuset: enabled

  --- Misc ---
  Veth pair device: enabled
  Macvlan: enabled
  Vlan: enabled
  File capabilities: enabled
  ```

## Templates

- To use legacy templates install the lxc-templates from AUR, however the template download is usually enough for most use cases:

```
yay -S lxc-templates
```

### Arch Container

- Create the container:

```
lxc-create -n arch.example.com -t download -- --dist archlinux --release current --arch amd64
```

  - **NOTE**: If using btrfs, lxc can take advantage of the subvoluming btrfs provides. To use btrfs, create the container with the following command:

    ```
    lxc-create -n arch.example.com -t download -B btrfs -- --dist archlinux --release current --arch amd64
    ```

  - **NOTE**: To display a list of available templates to download, use:

    ```
    lxc-create -n arch.example.com -t download
    ```

- Start the Arch container:

```
lxc-start -n arch.example.com
```

- Open a console to the container:

```
lxc-attach -n arch.example.com
```

## Debian Container

- Create the container:

```
lxc-create -n debian.example.com -t download -- -d debian -r bullseye -a amd64
```

  - **NOTE**: If using btrfs, lxc can take advantage of the subvoluming btrfs provides. To use btrfs, create the container with the following command:

    ```
    lxc-create -n debian.example.com -t download-B btrfs -- -d debian -r bullseye -a amd64
    ```

- *(Optional)* Add the Raspbian repository on top of stock Debian repos:

```
echo 'deb http://archive.raspbian.org/raspbian bullseye main contrib non-free rpi' >> /etc/apt
/sources.list.d/raspbian.list
echo 'deb-src http://archive.raspbian.org/raspbian bullseye main contrib non-free rpi' >> /etc
/apt/sources.list.d/raspbian.list
```

  - And add the Raspbian public signing key:

    ```
    wget https://archive.raspbian.org/raspbian.public.key -O - | sudo apt-key add -
    ```

  - Update the apt repository cache and upgrade any necessary files:

    ```
    apt update && apt upgrade
    ```

- *(Optional)* Add the Wolfram Alpha repository:

```
echo 'deb http://repository.wolfram.com/raspbian/ stable non-free' >> /etc/apt/sources.list.d/
wolfram.list
```

  - And add the Wolfram public signing key:

    ```
    apt-key adv –keyserver http://repository.wolfram.com/raspbian/raspbian@wolfram.com.gpg.pub
    -key –recv-keys 574FA74E5CBB4222
    ```

  - Update the apt repository cache and upgrade any necessary files:

    ```
    apt-get update && apt-get upgrade
    ```

  - Install wolfram:

    ```
    apt-get install wolfram-engine mathelxc-create -n kali.example.com -t kali-arm -- --releas
    e sana --mirror=http://archive.kali.org/kali --security=http://security.kali.org/kali-secu
    rity --packages=apt-utils,wget,debian-keyring,e2fsprogs,kali-defaults,kali-menu,parted,sud
    o,usbutilsmatica-fonts
    ```

## Ubuntu Container

- Create the container:

```
lxc-create -n ubuntu.example.com -t download -- -d ubuntu -r focal -a amd64
```

  - **NOTE**: If using btrfs, lxc can take advantage of the subvoluming btrfs provides. To use btrfs, create the container with the following command:

    ```
    lxc-create -n ubuntu.example.com -t download -B btrfs -- -d ubuntu -r focal -a amd64
    ```

## Fedora Container

- Create the container:

```
lxc-create -n fedora.example.com -t download -- -d fedora -r 34 -a amd64
```

- Edit the Fedora container config file:

```
nano /var/lib/lxc/fedora.example.com/config
```

  - And add/modify the following:

    ```
    lxc.network.type = veth
    lxc.network.flags = up
    lxc.network.link = br0
    lxc.network.ipv4 = 192.168.1.55/24
    lxc.network.ipv4.gateway = 192.168.1.1
    lxc.network.name = eth0
    lxc.autodev = 1
    lxc.pts = 1024

    lxc.rootfs = /var/lib/lxc/fedora.example.com/rootfs
    lxc.utsname = fedora.example.com
    lxc.arch = armvhf
    ```

- Start the containter:

```
lxc-start -n fedora.example.com
```

## ~~Kali Container~~

**NOTE**: Not sure if this still works, keeping around for posterity.

- Install debootstrap from AUR:

```
yaourt debootstrap
```

  - Install gnupg1 for keyring verification; make sure to edit the PKGBUILD and add **armv7h** to the arch parameter:

    ```
    gpg --keyserver pgpkeys.mit.edu --recv-keys 2071B08A33BD3F06
    yaourt gnupg1
    ```

  - Install debian-archive-keyring:

```
yaourt debian-archive-keyring
```

- Create a sana debootstrap script from the debian jessie script:

```
cp /usr/share/debootstrap/scripts/jessie /usr/share/debootstrap/scripts/sana
```

- Create a kali-arm lxc template script from the debian jessie template:

```
cp /usr/share/lxc/templates/lxc-debian /usr/share/lxc/templates/lxc-kali-arm
```

- Edit the kali-arm lxc template:

```
nano /usr/share/lxc/templates/lxc-kali-arm
```

  - And modify the following:

```
valid_releases=('squeeze' 'wheezy' 'jessie' 'stretch' 'sid' 'sana')

#...

    packages=\
ifupdown,\
locales,\
libui-dialog-perl,\
dialog,\
isc-dhcp-client,\
netbase,\
net-tools,\
iproute,\
kali-archive-keyring,\
kali-defaults,\
kali-menu,\
openssh-server

#...

debootstrap --verbose --variant=minbase --arch=$arch --keyring=/usr/share/keyrings/kali-archive-keyring.gpg
```

- Create the container:

```
lxc-create -n kali.example.com -t kali-arm -- --release sana --mirror=http://archive.kali.org/kali --security=http://security.kali.org/kali-security --packages=apt-utils,wget,e2fsprogs,parted,sudo,usbutils
```

  - **NOTE**: If using btrfs, lxc can take advantage of the subvoluming btrfs provides. To use btrfs, create the container with the following command:

```
lxc-create -n kali.example.com -t kali-arm -B btrfs -- --release sana --mirror=http://archive.kali.org/kali --security=http://security.kali.org/kali-security --packages=apt-utils,wget,e2fsprogs,parted,sudo,usbutils
```

- Edit the Kali container config file:

```
nano /var/lib/lxc/kali.example.com/config
```

  - And add/modify the following:

    ```
    lxc.network.type = veth
    lxc.network.flags = up
    lxc.network.link = br0
    lxc.network.ipv4 = 192.168.1.56/24
    lxc.network.ipv4.gateway = 192.168.1.1
    lxc.network.name = eth0
    lxc.autodev = 1
    lxc.pts = 1024

    lxc.rootfs = /var/lib/lxc/kali.example.com/rootfs
    lxc.utsname = kali.example.com
    lxc.arch = armvhf
    ```

- Copy over the resolv config from the host:

```
cp /etc/resolv.conf /var/lib/lxc/kali.example.com/rootfs/etc/
```

- Start the containter:

```
lxc-start -n kali.example.com
```

- Enable ssh at boot up:

```
arch-chroot /var/lib/lxc/kali.example.com/rootfs /bin/bash
/bin/systemctl enable ssh
exit
```

  - *(Optional)* Add bob the administrative user:

    ```
    arch-chroot /var/lib/lxc/kali.example.com/rootfs /bin/bash
    /usr/sbin/adduser bob
    /usr/sbin/usermod -aG sudo bob
    exit
    ```

  - Restart the container for the service to take effect:

    ```
    lxc-stop -n kali.example.com
    lxc-start -n kali.example.com
    ```

- Now connect to the kali contain via ssh:

```
ssh kali.example.com
```

  - *(Optional)* Install the desktop packages:

    ```
    apt-get install fonts-croscore fonts-crosextra-caladea fonts-crosextra-carlito gnome-theme-kali gtk3-engines-xfce kali-desktop-xfce kali-root-login lightdm network-manager network-manager-gnome xfce4 xserver-xorg-video-fbdev
    ```

  - *(Optional)* Install a few commomly used tools:

```
apt-get install aircrack-ng ethtool hydra john libnfc-bin mfoc nmap passing-the-hash sqlma
p usbutils winexe wireshark
```

- *(Optional)* Install a Kali Top10 meta-package:

```
apt-get install kali-linux-top10
```

- **NOTE**: If installing the kali-linux-all, kali-linux-wireless, or kali-linux-sdr meta packages, make sure to create a modified uhd-usrp2 sysctl file before installing the packages. If this is not done, the installation process will fail while installing the uhd-host package:

```
echo '# USRP2 gigabit ethernet transport tuning' >> /etc/sysctl.d/uhd-usrp2.conf
echo '#net.core.rmem_max=50000000' >> /etc/sysctl.d/uhd-usrp2.conf
echo '#net.core.wmem_max=1048576' >> /etc/sysctl.d/uhd-usrp2.conf
apt-get install kali-linux-sdr
```

## Autostart Containers

- Enable the lxc service at boot:

```
systemctl enable lxc@containername
```

## Device Passthru

One of my use cases involves passing a serial device to a specific container. LXC makes this task simple with the lxc-device command.

- Pass /dev/ttyACM0 to the arch.example.com containers /dev/ttyACM0:

```
lxc-device -n arch.example.com /dev/ttyACM0 /dev/ttyACM0
```

## Graphical Management

- Install virt-manager:

```
pacman -S virt-manager
```

1. Open virt-manager, then **Create a New Connection**.
2. Set the *connection type* to **LXC**, and set the *container type* to **Operating system container**.
3. Set the *OS root directory* to **/var/lib/lxc/arch.example.com/rootfs**.
4. Set a desired CPU and RAM amount.
5. Set a name for the container.

After the container is configured it will automatically start.

## Shared Package Cache for Arch Containers

**NOTE**: I found this trick to be incredibly useful by allowing me to update the host's packages, then use the downloaded packages to cut down on download time for updating the Arch containers; since many of the packages are the same between updates. Why download more than once if you don't have to, right?

### Shared Per Container

- Edit the arch linux container config file:

```
nano /var/lib/lxc/arch.example.com/config
```

- And add the following to set the containers package cache to the host's package cache directory:

  ```
  # Share host package cache with this container
  lxc.mount.entry = /var/cache/pacman/pkg var/cache/pacman/pkg none bind 0 0
  ```

- **NOTE**: I needed to attach to the container and refresh the pacman database list:

  ```
  lxc-attach -n arch.example.com
  pacman -Sy
  exit
  ```

### Shared With All Arch Containers

- Edit the common arch linux container config file:

```
nano /usr/share/lxc/config/archlinux.common.conf
```

- And add the following to set the containers package cache to the host's package cache directory:

  ```
  # Share host package cache with all containers
  lxc.mount.entry = /var/cache/pacman/pkg var/cache/pacman/pkg none bind 0 0
  ```

## Resources

- https://s3hh.wordpress.com/2011/05/17/lxc-containers-on-a-host-with-wireless/
- https://wiki.archlinux.org/title/Linux_Containers
- https://www.raspbian.org/RaspbianFAQ#Can_I_mix_packages_from_the_Debian_repositories_with_Raspbian.3F
- https://www.raspbian.org/RaspbianRepository
- https://www.raspberrypi.org/forums/viewtopic.php?f=94&t=61509
- https://github.com/andrius/build-raspbian-image/issues/1
- http://docs.kali.org/kali-on-arm/install-kali-linux-arm-raspberry-pi
- http://docs.kali.org/development/kali-linux-arm-chroot
- http://docs.kali.org/general-use/kali-linux-sources-list-repositories
- https://forums.kali.org/showthread.php?18079-Public-key-error
- https://github.com/offensive-security/kali-arm-build-scripts/blob/master/rpi2.sh
- http://www.kali.org/new/kali-linux-metapackages/

### History

**#1 - 10/22/2015 10:58 AM - Daniel Curtis**

*- Subject changed from Creating Linux Containers on Arch Linux for Raspberry Pi to Creating Linux Containers on Arch Linux for Raspberry Pi 2*

*- Status changed from New to In Progress*

*- % Done changed from 0 to 10*

**#2 - 10/23/2015 11:32 AM - Daniel Curtis**

*- Description updated*

*- Status changed from In Progress to Resolved*

*- % Done changed from 10 to 100*

**#3 - 10/24/2015 08:03 PM - Daniel Curtis**

*- Description updated*

**#4 - 10/26/2015 04:42 PM - Daniel Curtis**

*- Description updated*

**#5 - 11/05/2015 05:51 PM - Daniel Curtis**

*- Description updated*

**#6 - 11/06/2015 04:44 PM - Daniel Curtis**

*- Description updated*

**#7 - 11/06/2015 05:35 PM - Daniel Curtis**

*- Description updated*

**#8 - 11/09/2015 01:13 PM - Daniel Curtis**

*- Description updated*

*- Estimated time set to 2.00 h*

**#9 - 11/09/2015 02:33 PM - Daniel Curtis**

*- Description updated*

**#10 - 11/09/2015 02:43 PM - Daniel Curtis**

*- Description updated*

**#11 - 11/09/2015 09:05 PM - Daniel Curtis**

*- Description updated*

**#12 - 11/14/2015 08:16 AM - Daniel Curtis**

*- Description updated*

*- Estimated time changed from 2.00 h to 3.00 h*

**#13 - 11/14/2015 09:08 AM - Daniel Curtis**

*- Description updated*

**#14 - 11/14/2015 09:40 AM - Daniel Curtis**

*- Description updated*

**#15 - 11/14/2015 11:39 AM - Daniel Curtis**

*- Subject changed from Creating Linux Containers on Arch Linux for Raspberry Pi 2 to Create Linux Containers on Arch Linux for Raspberry Pi 2*

*- Description updated*

**#16 - 11/14/2015 05:15 PM - Daniel Curtis**

*- Description updated*

**#17 - 11/15/2015 10:13 AM - Daniel Curtis**

*- Description updated*

**#18 - 11/16/2015 03:56 PM - Daniel Curtis**

*- Description updated*

**#19 - 11/27/2015 03:49 PM - Daniel Curtis**

*- Status changed from Resolved to Closed*

**#20 - 12/06/2015 12:40 PM - Daniel Curtis**

*- Description updated*

*- Category changed from Jails / Containers to Jails / Container*

*- Target version changed from Arch Linux to Arch Linux*

*- Project changed from Raspberry Pi to GNU/Linux Administration*

*- Subject changed from Create Linux Containers on Arch Linux for Raspberry Pi 2 to Linux Containers on Arch Linux*

**#21 - 12/11/2015 05:02 PM - Daniel Curtis**

*- Description updated*

**#22 - 12/16/2015 12:00 PM - Daniel Curtis**

*- Description updated*

**#23 - 12/16/2015 01:17 PM - Daniel Curtis**

*- Description updated*

**#24 - 01/03/2016 06:58 PM - Daniel Curtis**

*- Description updated*

**#25 - 01/09/2016 11:08 AM - Daniel Curtis**

*- Description updated*

**#26 - 01/09/2016 11:41 AM - Daniel Curtis**

*- Description updated*

**#27 - 02/22/2016 08:56 PM - Daniel Curtis**

*- Description updated*

**#28 - 07/07/2016 08:26 PM - Daniel Curtis**

*- Description updated*

**#29 - 07/07/2016 10:54 PM - Daniel Curtis**

*- Description updated*

**#30 - 08/06/2016 10:23 AM - Daniel Curtis**

*- Description updated*

**#31 - 01/26/2022 11:33 AM - Daniel Curtis**

*- Description updated*

**#32 - 01/27/2022 12:08 PM - Daniel Curtis**

*- Description updated*