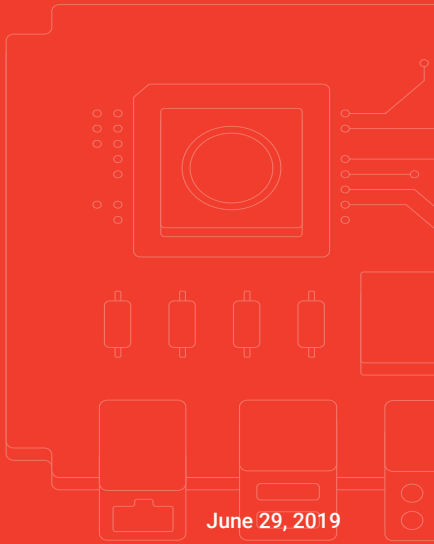
Zagreb, Cloud analysis

Container technologies

Davor Popović · Marko Ratkaj



sartura

June 29, 2019

LXC/LXD

- LXD = container manager:
  1. Run containers
  2. Update containers
  3. Clustering support
  4. Install different Linux distributions inside containers
  5. Manage container resources, like storage volumes and networks

- Procedure when working with LXD:
  1. Install lxd
  2. Setup and configure LXD networking and storage
  3. Create your first container
  4. Launch the container

- First, initialize the LXD daemon

```
lxd init
```

- Setting up basic environment for daemon to work in
  - Storage, basic networking, …
- To verify that everything went ok and everything is configured

```
lxc init
lxc info | more
```

Creating the first container

- Two possibilities
  1. Provide manually built rootfs from another distribution – this is usually created from existing rootfs or with some other method. Probably takes additional effort but allows building from scratch.
  2. Use prepared images from upstream – idea is that LXD downloads already-prepared images and uses them for containers. It is easier but it has to be additionally modified once set up if needed.

- What is available on upstream?

```
lxc image list images
```

- To find a specific upstream image based on the distribution use simple `grep` :

```
lxc image list images: | grep -i 'debian'
```

- Alternatively, you can also use

```
lxc image list images: 'debian'
```

- Create a new image and launch the container:

```
lxc launch images:<UPSTREAM_ALIAS> <LOCAL_ALIAS>
```

- Manual container creation
  - Prepare rootfs
- Each step must be configured manually
  - Create container rootfs (compressed)
  - Create container metadata (basic info about the container)
  - Create container profile (basic configuration of the container)

- Create rootfs directory where everything will be placed:

```
cd ~
mkdir rootfs
```

- Here, place metadata and profile file
- Both of these are written either in JSON or in YAML

- Minimal metadata template file

```
vim metadata.yaml
```

```
1  architecture: "aarch64"
2  creation_date: 1554382805      # mandatory and must be valid. Each container must have unique.
       Take this value: date +%s
3  properties:
4          architecture: "aarch64"
5          description: "Example of Gentoo virtual router"
6          os: "Gentoo Linux"
7          release: "0.1"
8          variant: "Custom"
```

- Prepare container profile as well

```
config: {}
description: Gentoo LXD profile
devices:
  eth0:
    name: eth0
    nictype: macvlan
    parent: eth1^^I^^I^^I^^I#this can vary, depending on how is the interface named on host (
     enpXXX, ethXXX, enoXXX...)
    type: nic
  root:
    path: /
    pool: workstation-pool
    type: disk
name: default
```

- Next step is to compress both rootfs and metadata

```
tar czf rootfs.tar.gz rootfs
tar czf metadata.tar.gz metadata.yaml
```

- Both of these need to be imported into the LXD daemon to be ready to use:

```
lxc image import rootfs.tar.gz metadata.yaml –alias my-image
```

- If everything went ok, the image should be listed in the internal list of images by the LXD daemon

```
lxc list
```

- At this point, we only have image – not a container as such
- Before starting a container, our recommendation is to prepare a profile for the container
- We have already written the profile, now it also needs to be imported into the daemon for usage
- One profile can be applied to multiple containers

- First step is to create a profile:

```
lxc profile create my-profile
```

- This creates an internal profile called `my-profile` in LXD but it still has no values/parameters set
- To set it up, the profile can be directly redirected from text file to profile inside the daemon (on *nix systems this is done with <,> symbols):

```
lxc profile edit my-profile < my-profile.yaml
```

- To check if the profile has been applied and has no errors:

```
lxc profile list
```

- The container can now be started:

```
lxc start my-container
```

- At this point, any program can be started inside the container by calling `exec`, e.g. start `ping` with few packets

```
lxc exec my-container -- /bin/ping 8.8.8.8 -c2
```

- In order to enter inside the container, the shell can be executed
- From the container's shell it is possible to do almost any operation as on the host machine – the user experience remains the same

```
lxc exec my-container -- /bin/bash
```

Setting up network inside the container

- Practical example – setting up the network inside the container
- As defined in the container profile, the interface from the container is connected directly to a physical interface on the host machine with `macvlan` interface
- `macvlan` creates a new interface with a different MAC address than the host one and allows traffic to go directly through (as opposed to a bridge where it has to hit the bridge first)

- In theory, there is nothing wrong with this configuration
- In practice, network has to be configured inside the container as well
- The user can either set up static IP on the inside interface or set dynamic IP (meaning that the IP on the container interface will be offered by someone else – DHCP server running somewhere in the network)

- How?
  - `systemd`
  - Daemon in role of PID 1 – master process, initial process from which all other processes are spawned
  - One of the domains directly under `systemd` control is networking
  - As any other program, `systemd` and its components are configured with different configuration files located under `/etc/system/(network)`

- Listing out `/etc/systemd/network` might show that it is empty so a new file containing network configuration must be created
- A good practice is to name the file `<file_name>.network`
- This file will define the following:
  - Match the given interface
  - Assign it with IPv4 address from a DHCP server

- Create a file

```
vim /etc/systemd/network/eth0.network
```

- Add the following:

```
[Match]             # Match this interface
Name=eth0

[Network]           # Assign IPv4 address from a DHCP server
DHCP=ipv4

[DHCP]
RouteMetric=10
```

- Restart systemd networking service

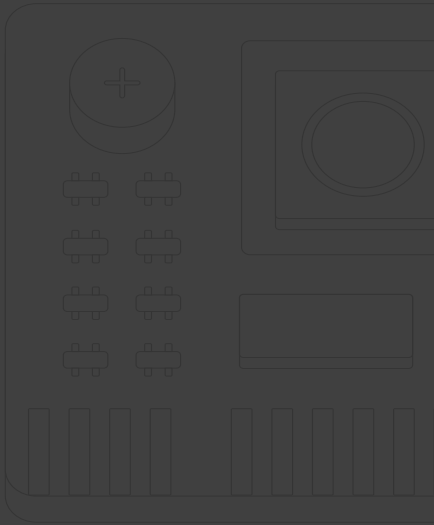
```
systemctl restart systemd-networkd
```

- At this point, on the `eth0` interface an IP address should appear and it should be from the same subnet as the IP address offered on the physical interface of the host
- Try pinging the Internet

```
ping 8.8.8.8
```

Container technologies

davor.popovic@sartura.hr · **marko.ratkaj@sartura.hr**



info@sartura.hr · www.sartura.hr