

# Deployment & Virtualization

---

Joseph Chazalon, Clément Demoulins {`firstname.lastname@lrde.epita.fr`}

October'19

EPITA Research & Development Laboratory (LRDE)

## How to manage multi docker containers

---

Docker compose can manage/configure multiple containers on a single host or more recently on a docker swarm cluster. You can configure :

- services (running containers)
- ports
- networks to connect services together
- volumes

Use cases :

- Manage containers on a production server (?)
- Avoid shell script to start your container

Hello-world example:

```
version: '3'
services:
  hello:
    image: hello-world
```

## Docker compose example 2/2

```
$ docker-compose up
Creating network "tmp_default" with the default driver
Pulling hello (hello-world:)...
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:b8ba256769a0ac28dd126d584e0a2011cd2877f3f76e093a7ae560
Status: Downloaded newer image for hello-world:latest
Creating tmp_hello_1 ... done
Attaching to tmp_hello_1
hello_1 |
hello_1 | Hello from Docker!
hello_1 | This message shows that your installation appears to be working
hello_1 |
(...)
hello_1 |
tmp_hello_1 exited with code 0
```

## Docker compose more real life example

Docker compose example from

<https://docs.docker.com/compose/gettingstarted/>

```
verison: '3'
```

```
services:
```

```
  web:
```

```
    build: .
```

```
    ports:
```

```
      - "5000:5000"
```

```
    volumes:
```

```
      - ./code
```

```
  redis:
```

```
    image: "redis:alipne"
```

- Manage docker cluster
- Integrated to docker (docker swarm command)
- Internal load balancer
- Docker compose can deploy services directly into a docker swarm cluster
- Easy and lightweight setup
- No dashboard but you can use **<https://www.portainer.io>**

Google version of docker swarm with :

- Almost all docker swarm functionalities
- Auto-scaling
- Large community
- Complex installation/configuration
- Incompatible with docker CLI and compose tools



## Containers alternatives

---

# LXC (Linux Container)

- Use case : lightweight VM
- Used to be the runtime backend of docker before containerd/runc
- First linux container tool
- Doesn't manage images

Example of a config file:

```
lxc.rootfs.path = /var/lib/lxc/playtime/rootfs
lxc.uts.name = playtime
lxc.arch = x86_64
lxc.include = /usr/share/lxc/config/common.conf
lxc.net.0.type = veth
lxc.net.0.link = br0
lxc.net.0.flags = up
lxc.net.0.name = eth0
lxc.net.0.hwaddr = ee:ec:fa:e9:56:7d
lxc.net.0.ipv4.address = 192.168.0.3/24
lxc.net.0.ipv4.gateway = 192.168.0.1
```

- Modern version of LXC based on systemd
- Low level command from systemd : `systemd-nspawn`, `machinectl`
- Doesn't manage images : you have to manually initialize your root filesystem with tools like `debootstrap` or `pacstrap`
- Support for OCI runtime (Open Container Initiative)

**`systemd-nspawn --machine=cbuster --boot`**

- Compatibility with docker image format
- Try to be more secure by default
- Use systemd-nspawn to run container
- Actually the only alternatives to docker

Hello-world example :

```
rkt --insecure-options=image run docker://hello-world
```