Linux Containers

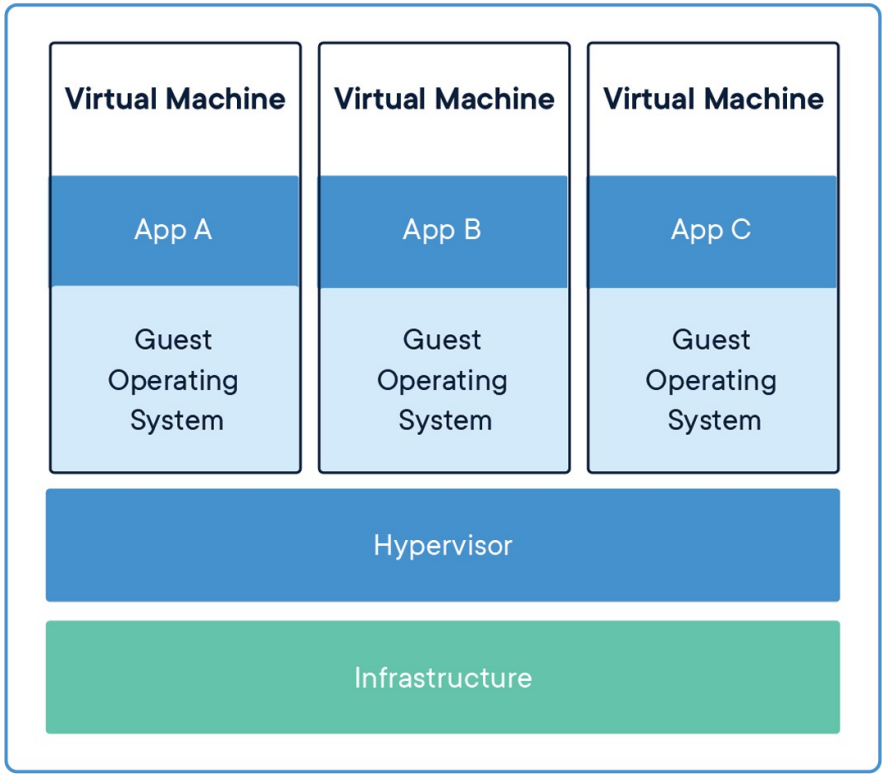
From chroot to Docker

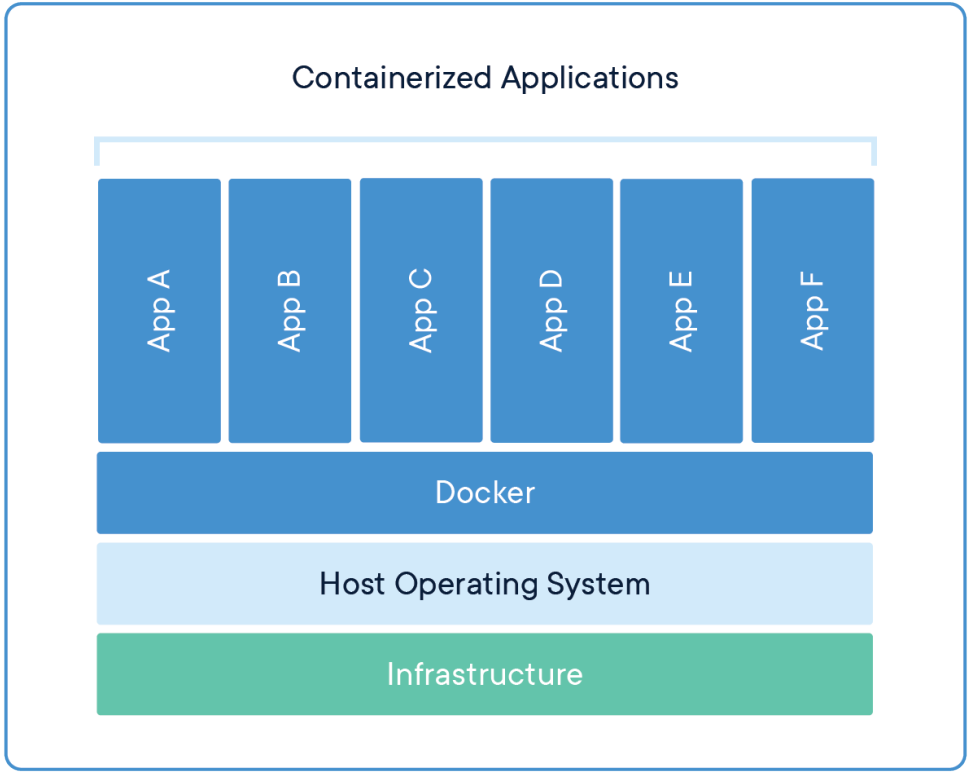
Ondřej Votava

Content

- Virtual Machine x Container
- Containers Evolution
- chroot
- Open VZ
- LXC/LXD
- systemd-nspawn
- Docker

Virtual machine x Container





Source of images: www.docker.com

Virtual machine x Container

Virtual Machine

- **Virtual Hardware**
  - CPU
  - Memory
  - Network
- **Any OS**
- **“Big” overhead**

Container

- **Isolated Processes**
- **Limited Resources**
  - CPU
  - Memory
  - Network
- **Linux (Unix, Win) Only**
- **“Small” overhead**

Virtual machine x Container – example HTTP server

Virtual Machine

- **2 cores**
  - shared / overcommit
- **2 GB RAM**
  - for both OS and HTTP server
- **10 GB storage**
  - 2 – 4 GB for OS, swap etc.
  - 10 MB HTTP server, rest for application data and logs

Container

- **2 cores**
  - shared / overcommit
- **2 GB RAM**
  - only for HTTP server
- **10 GB storage**
  - 10 MB HTTP server, rest for application data and logs

Containers Evolution [1]

- **1979 – Unix v7 – chroot**
  - File access segregation for each process
- **2000 – FreeBSD Jails**
  - Files system, network, process tree isolation
- **2004 – Solaris Zones**
- **2005 – OpenVZ, LXC**
- **2007 – Control Groups (cgroups)**
- **2008 – LXC with cgroups**
- **2013 – Docker**
- **2014 – LXC 1.0 – unpriviledged containers support**

[1] https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016


Linux Containers

Change root (chroot)

- **File system isolation**
  - Processes cannot acces files out of the new root tree
  - Shared sys and proc files / security flaws when root access allowed
- **Shares host networking**
- **OpenSSH server access restriction** [1]

```
chroot option newroot [command [args]…]
  --groups=groups
  --userspec=user[:group]
```

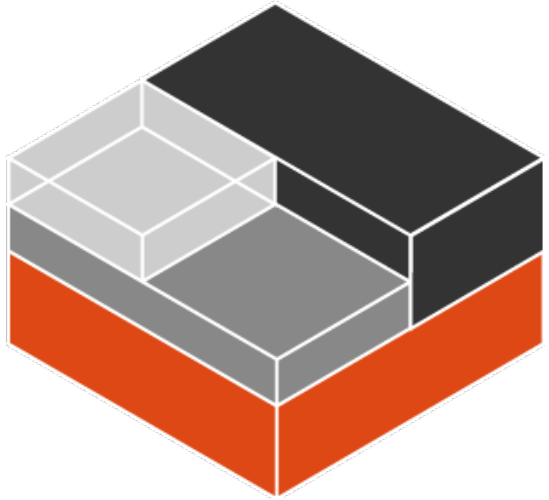
[1] https://www.tecmint.com/restrict-ssh-user-to-directory-using-chrooted-jail/

Open VZ (Open Virtuzzo)

- **Custom kernel – based on RHEL7**
- **Specific distro**
- **Isolation & limitation**
  - File system
  - Process tree
  - Network
- **Supports live migration**
- **No active development – last release 2016**

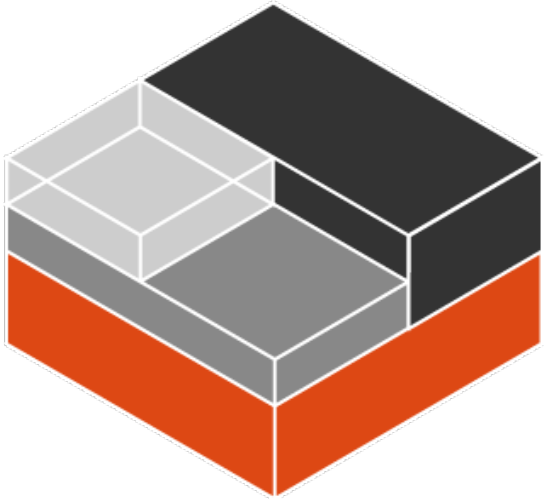
LinuX Containers (LXC)

- Kernel namespaces (ipc, uts, mount, pid, network and user)
- Apparmor and SELinux profiles
- Seccomp policies
- Chroots (using pivot_root)
- Kernel capabilities
- CGroups (control groups)



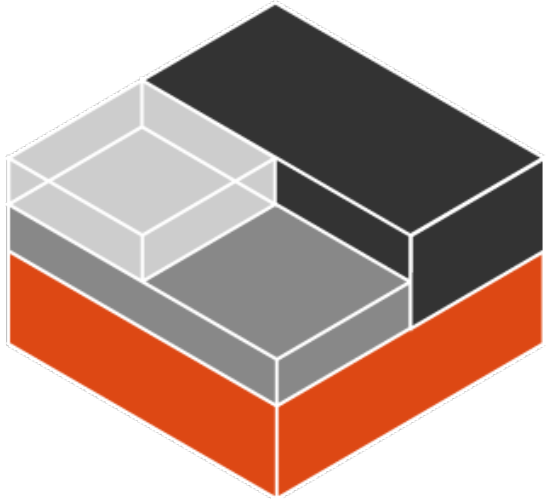
LinuX Containers (LXC)

- **liblxc**
- **API libraries**
  - python3, lua, Go, ruby, ...
- **Set of CLI tools**
- **Set of templates**
- **LXCFS**
  - Solves systemd inside unpriviledged containers



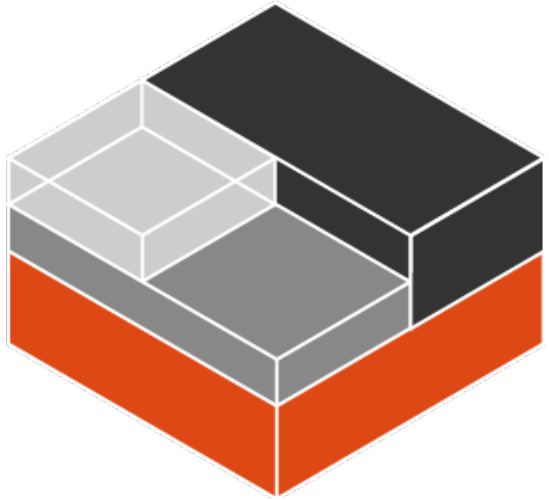
LinuX Containers (LXC) – Networking

- **Type**
  - **none** – shared with host
  - **empty** – loopback only
  - **veth** – virtual ethernet
  - **vlan** – support for vlan on a host's NIC
  - **macvlan** – new MAC address on host's NIC
  - **ipvlan** – new IP address on host's NIC



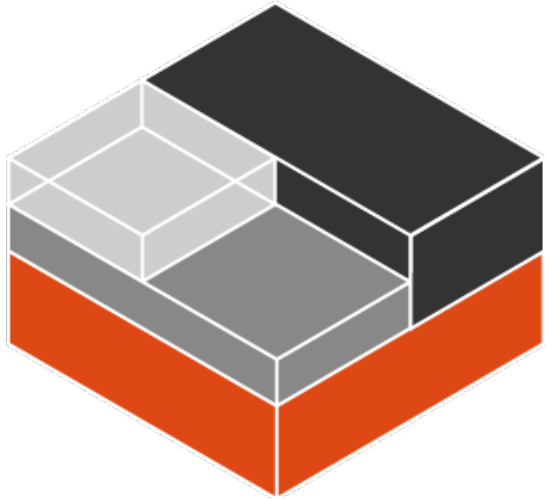
LinuX Containers (LXC) – Storage

- **Backing Storage Types**
  - **btrfs, zfs, rbd**
  - **dir** – default
  - **lvm, loop** – binary images
  - **overlay** – for clones (snapshot)
- **Ephemeral – remove after stop**



LinuX Containers (LXC) – Example

- `lxc-create -t ubuntu -n my-ubuntu`
- `lxc-start -n my-ubuntu -d`
- `lxc-ls -f`
- `lxc-attach -n my-ubuntu`
- `lxc-stop -n my-ubuntu`
- `cat /var/lib/lxc/my-ubuntu/config`
- `lxc-destroy -n my-ubuntu`



systemd-nspawn

- **Container System for systemd based distros**
- **Similar to chroot**
  - Full filesystem virtualization
  - Independent process tree
  - IPC subsystems and host and domain name independent
- **Containers may run as a systemd service**
  - nspawn@.service template

[● ◀] systemd

systemd-nspawn – Networking

- Private – should container isolate from host’s network
- VirtualEthernet
- MACVLAN
- IPVLAN
- Zone – isolate containers from each other
- Port – allow port forwarding from host for private networking

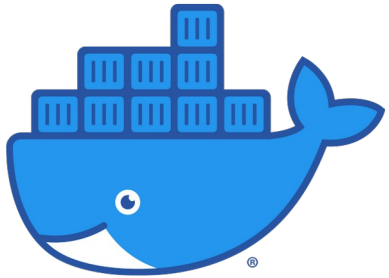
systemd-nspawn – Storage

- **Binary images**
  - Raw
  - Block device
  - MBR, GPT, EFI
- **Ephemeral**
- **Templates**
  - BTRFS based snapshots
- **Volatile**

systemd-nspawn – Example

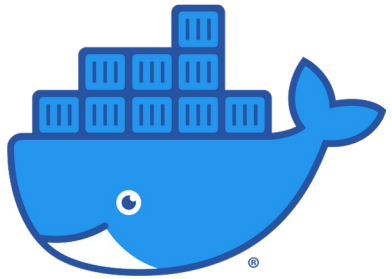
- `debootstrap focal /var/lib/machines/my-ubuntu`
- `systemd-nspawn -M my-ubuntu passwd`
- `systemd-nspawn -M my-ubuntu --hostname my-ubuntu`
- `systemd-nspawn -M my-ubuntu --boot`
- `machinectl list-images`
- `machinectl list`

Docker



Docker

- **Single process virtualization (service)**
- **Ecosystem for application distribution**
  - **Docker Hub**
    - Free/paid storage for images ready to use
  - **Dockerfile**
    - Yaml description of how to create a service
  - **Docker Compose**
    - Applications with more services
  - **Docker Swarm** – orchestration tool



Docker Engine

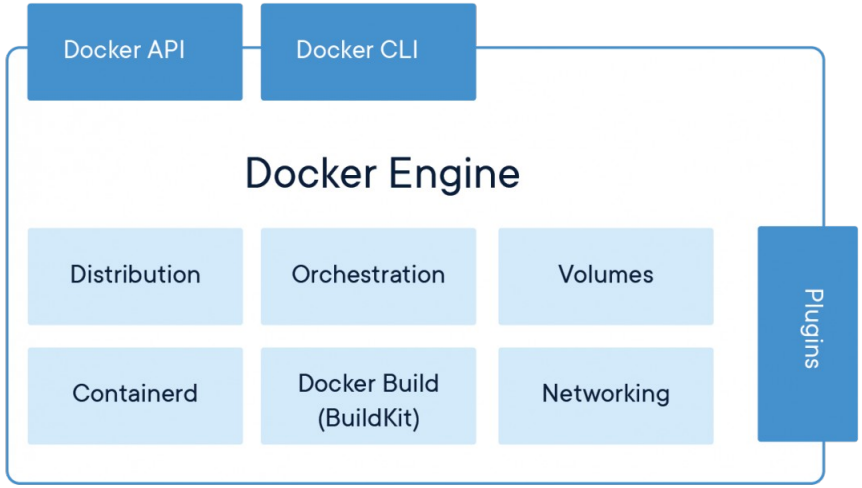


Image from https://www.docker.com/products/container-runtime

Dockerfile

- **Description of a service – YAML text format**
- **Base image**
  - Static linked binary
  - Prepared Linux distribution – popular Alpine, Ubuntu, Debian, CentOS, ...
  - Any Docker image
- **Layers**
  - Executing a command (add, copy, run) creates a new layer
  - Final image contains many layers – try to minimize by merging commands
- **Multi-stage builds**

Dockerfile – Example

```
FROM ubuntu:latest

RUN apt-get update && \
    apt-get install -y nginx && \
    apt-get autoremove && \
    rm -rf /var/lib/apt/lists/*

COPY my-awesome-app /var/www/html
```

```
FROM nginx:latest

COPY my-awesome-app /var/www/html
```

Docker – Run Container

- **Start the nginx container in background, mount data and configuration and expose the host's port 8080**

```
docker run --name my-container \
   -v /path/to/data:/var/www/html/data \
   -v /path/to/config/nginx.conf:/etc/nginx/nginx.conf:ro \
   -d -p 8080:80 nginx:latest
```

Docker – Networking

- **Implicit Docker network**
  - Masquerade
  - Port Forwarding
- **Custom networks**
  - Groups of containers share one network
- **Modes**
  - bridge, host, overlay, macvlan, ipvlan, none
  - container

Docker – Networking

- **IPv6**
  - Not enabled by default
  - Customization on host needs to be done
- **Firewall**
  - Docker add PREROUTING rules to firewall
  - These are always executed BEFORE rules added by e.g. UFW
  - IP addresses in rules are DYNAMIC, i.e. the order of starting services depends

Docker Compose

- Starting containers from command line is tedious
- Compose file describes all options for command line in YAML based text file
- More than one container can be described
- Implicit private network between containers established
- Exposed ports have to be explicitly defined
- Container names are resolvable from all containers within Compose file

Docker Compose – Example

```
version: "2.4"
services:
  web:
    build:
      dockerfile: webapp/Dockerfile
      context: webapp/.
    image: registry.example.org:443/webapp
    volumes:
      - webapp:/var/www/html
    ports:
      - 8080:80
    environment:
      MYSQL_SERVER: mysql
```

Docker Compose – Example cont.

```
  mysql:
    image: mysql:latest
    volumes:
      - mysql_data
    environment:
      PASSWORD: mysql
      DATABASE: webapp
volumes:
  - mysql_data
```

Docker Compose – Run a service

- docker-compose build
- docker-compose up
- docker-compose push
- docker-compose pull
- docker-compose up -d
- docker-compose stop
- docker-compose down

Docker Orchestration

- **Swarm**
  - Embedded in docker engine
  - Provides execution, replication, load balancing
- **Kubernetes**
  - Complex ecosystem for container management
  - Docker runtime support deprecated since Dec. 2020
  - Supports Open Container Initiative (OCI) images
  - Containerd and CRI-O as runtime


Questions?