Virtualization on Linux & Docker

2019/3/25 IJen

Outline

- Intro to components in Linux
  - Libvirt
  - Storage
  - Network
- Docker
  - Container v.s. Virtual Machine
  - Concept
- Lab

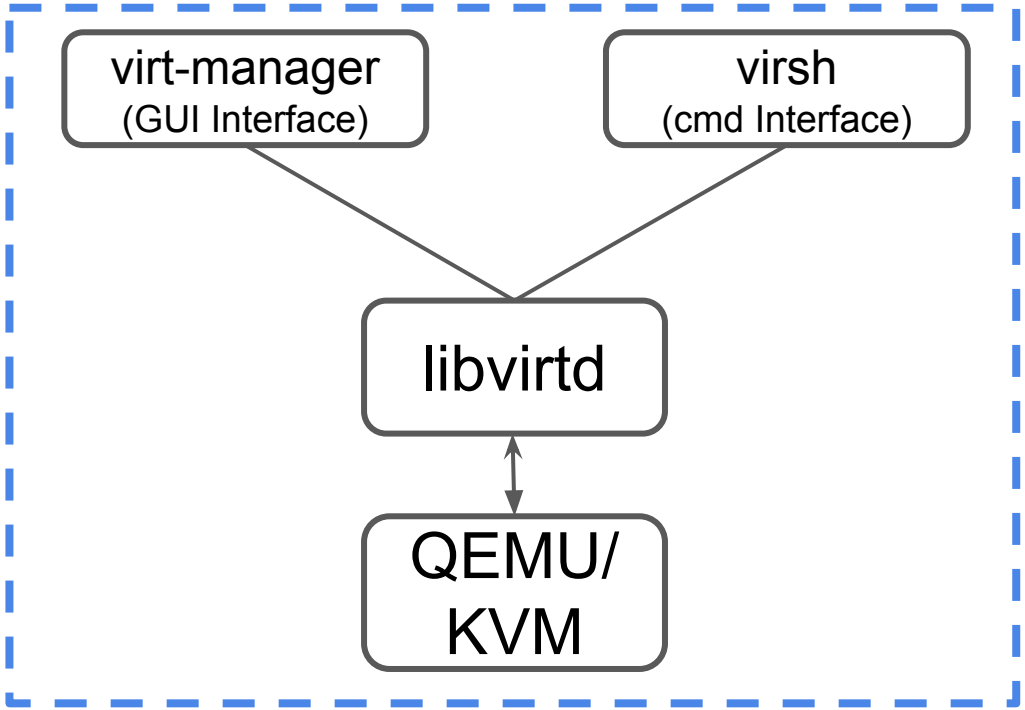
Intro to components in Linux

Libvirt (from ArchWiki)

Libvirt is collection of software that provides a convenient way to manage virtual machines and other virtualization functionality, including

- storage and network interface
- a long term stable C API, a daemon (libvirtd)
- a command line utility (virsh)
- multiple different virtualization providers/hypervisors (KVM/QEMU, Xen...)

Virtualization on Linux (localhost)



Storage

- raw
  - plain file, default
  - optimal performance
  - only very basic features are available
- qcow2
  - format of virtual disk image for QEMU
  - support copy on write
  - support resizing
  - support snapshot
  - And so on

Network

- NAT
  - default
  - DHCP/DNS
- Bridge
  - Direct accessible from outside
  - Does not support wireless network
  - It is administered using brctl command on Linux

Docker

Container v.s. Virtual Machine

Container

- No guest OS, share kernel with host.
- In principle, one container is for one process.
- Start time: < 1 sec
- Resource required: same as a process.
- Worse isolation.

Virtual Machine

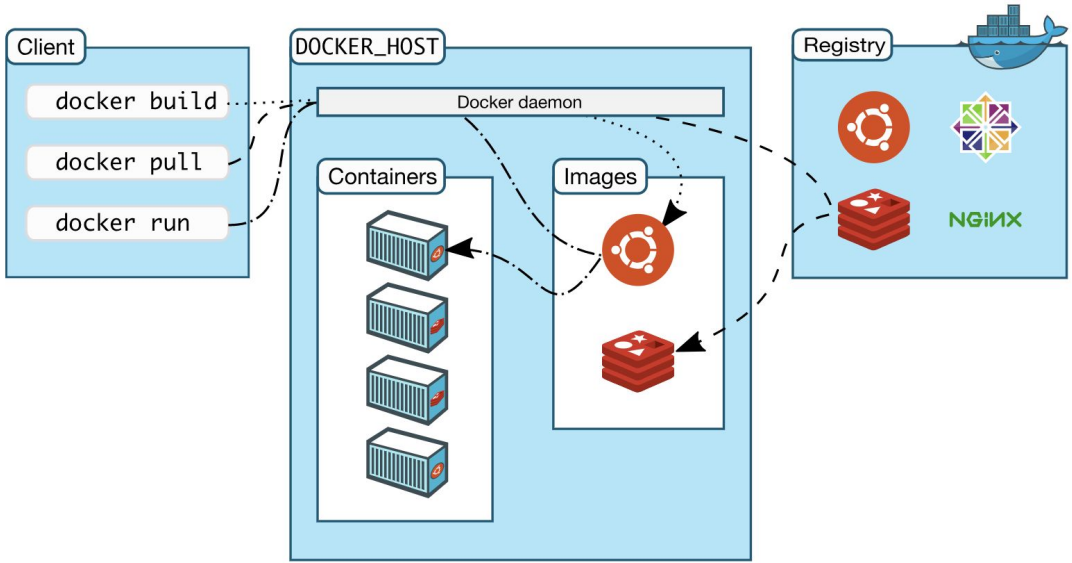
- Guest OS is required!
  Can have kernel different from host.
- One VM is for one OS, along with many processes.
- Start time: > 1 min
- Resource required: same as a OS.
- Better isolation

Concept

- Image
  - Read-only template
  - Like a blueprint
  - Images are created with the `build` command, and they'll produce a container when started with `run`.
- Dockerfile
  - To build your own image, you create a Dockerfile with a simple syntax for defining the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image.

Concept

- Docker Engine
  - Server (dockerd), REST API, CLI Client (docker)
- Architecture



Get Started

- Create image
  - Pull from online
  - Write your own dockerfile
- Build
- Run!

Lab

Lab

- Goal :
  - Build and run a docker image that contains the required files and packages
- Requirements :
  - Have apache (A web server, call “httpd” on CentOS)
  - Set up a simple website containing your student ID
  - Show TA your website and dockerfile

Create index.html

```
mkdir webserver
```

```
vim webserver/index.html
```

- Place your student ID
- And write anything you want

```
vim webserver/Dockerfile
```

- Hint is at next page

Dockerfile

`FROM centos:latest`	To specify the base image. A Dockerfile must start with a `FROM` instruction.
`RUN yum install ...`	To execute commands in a new layer on top of the current image and commit the results.
`COPY $src /var/www/html/`	To copy a local file or directory from your host into the Docker image itself.
`CMD [“”,...]`	To provide defaults for an executing container. There can only be one `CMD` instruction in a Dockerfile
`EXPOSE $port`	To informs Docker that the container listens on the specified network ports at runtime

Dockerfile (other instruction)

ADD	Similar to `COPY`, but allow to use a URL or extract a tar file as source file
VOLUME	To create a mount point and marks it as holding externally mounted volumes from native host or other containers.
USER	To set the user name (or UID) and optionally the user group (or GID) to use when running the image and other instructions. Default will be root.
WORKDIR	To set the working directory for following instructions. If the WORKDIR doesn't exist, it will be created
...	

Build and Run

```
docker build /webserver/ -t webserver

docker run –d –p 1234:80 webserver
```