Docker vs another container solutions

Overview

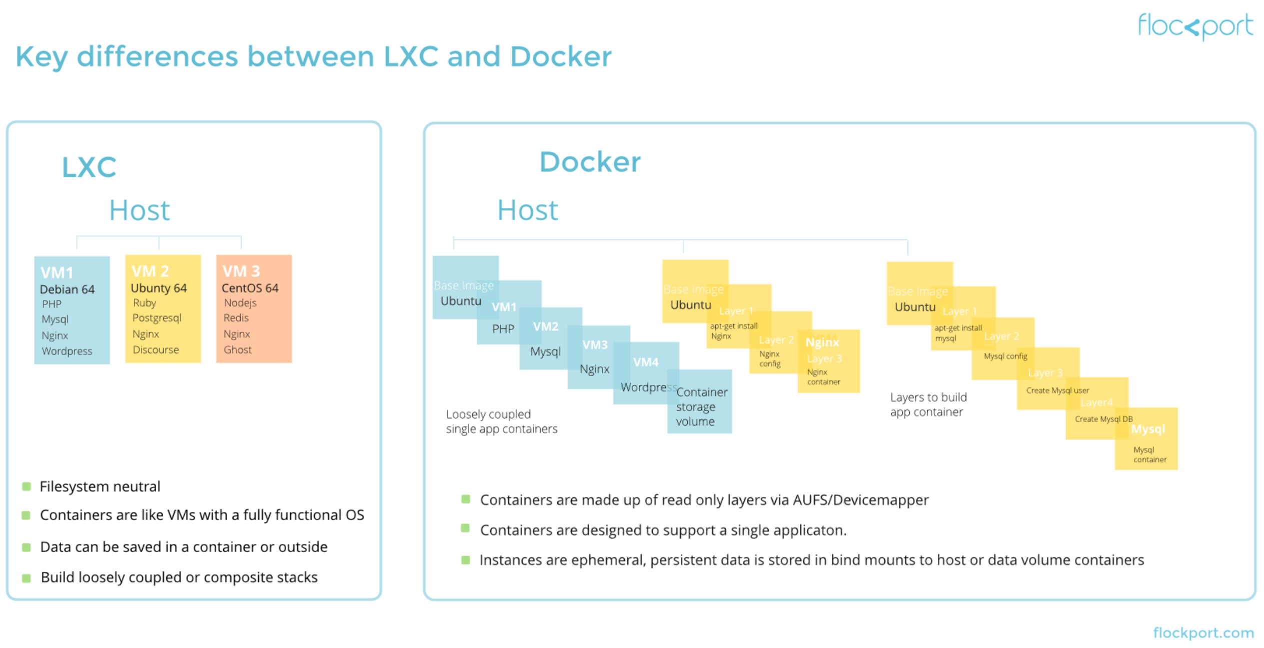
Container solution means approach where isolation targeted for application process inside of child environment. Child environment of container reuse kernel and environment setup from parent environment with ability to customise child environment with additional items. You can achieve with container solution same customisation of child environment as with VM's, but this is optional feature, wile for VM's its mandatory to setup each VM from scratch. Most container solutions come with declarative format of child environment definition.

Virtual machine solution means approach that allows to have completely new child environment inside of parent environment. Some vendors also called theirs approach as containers, but actually its a lightweight virtual machines. VM's have similar concept: you need to setup and configure full OS inside of child environment. Technically, each VM also can be described using scripts (Chef, Puppet, Ansible or SALT tools).

Here is a good description of differences and pros vs cons http://stackoverflow.com/a/40189938

Containers vs VM's

	Containers	Lightweight VM	VM
Description	• uses host kernel • uses host OS setup • uses isolation layers to enhance OS setup per container	• use host kernel • requires full environment setup on top of existing kernel per each VM	• require need new kernel setup • requires new kernel and new environment setup per each VM
Advantages	• lightweight process isolation • fast start up time • small footprint • better for application portability • declarative configuration language	• medium hardware resources isolation • stronger security isolation • better for machine portability	• good hardware resources isolation • stronger security isolation • better for machine portability • can execute any kernel inside of VM
Disadvantages	• medium hardware resources isolation • week security isolation • tied to host kernel	• medium footprint • medium startup time • tied to host kernel • need third party tools to configure in declarative way	• large footprint • long startup time • need third party tools to configure in declarative way
Projects	• Docker • rkt	• LXC/LXD • OpenVZ	• XEN • KVM


flockport
Key differences between LXC and Docker
LXC
Host
VM1 Debian 64 PHP Mysql Nginx Wordpress
VM 2 Ubunty 64 Ruby Postgresql Nginx Discourse
VM 3 CentOS 64 Nodejs Redis Nginx Ghost
Filesystem neutral
Containers are like VMs with a fully functional OS
Data can be saved in a container or outside
Build loosely coupled or composite stacks
Docker
Host
Base Image Ubuntu
VM1 PHP
VM2 Mysql
VM3 Nginx
VM4 Wordpress
Container storage volume
Loosely coupled single app containers
Base Image Ubuntu
Layer 1 apt-get install Nginx
Layer 2 Nginx config
Nginx Layer 3 Nginx container
Base Image Ubuntu
Layer 1 apt-get install mysql
Layer 2 Mysql config
Layer 3 Create Mysql user
Layer4 Create Mysql DB
Mysql Mysql container
Layers to build app container
Containers are made up of read only layers via AUFS/Devicemapper
Containers are designed to support a single applicaton.
Instances are ephemeral, persistent data is stored in bind mounts to host or data volume containers
flockport.com

Container solution comparison

Comparison table, describes pros/cons of different kinds of OS-level virtualisation solutions: https://en.wikipedia.org/wiki/Operating-system-level_virtualization#Implementations

	Docker	Rocket (rkt)	Mesos Universal Container Runtime
Image definition	• docker file format	• aci file format	• docker file format • oci, appc file formats
Image distribution	• Docker registry	• Quay registry (docker compatible)	• Docker registry
Security purpose	Uses docker engine priveleges	Positioned as more secured than Docker in terms of privileges separation	Can't find info
Companies using	• Netflix • The New York Times • PayPal • Splunk • The Washington Post • Swisscomm • GE • Uber • Ebay • Shopify • Spotify • New Relic • Yelp	• Bla Bla Car • XOOM • Viacom	• AirBnB • Verizon Most companies use Mesos as resource sheduler, its hard to find who uses containers solution
Orchestrators compatibility	Kubernetes, Mesosphere Marathon, Swarm, Nomand, Cloud Foundry's Diego,	Kubernetes, Nomand	Mesosphere Marathon
License	Apache 2.0	Apache 2.0	Apache 2.0
Code base activity	• 1761 contributors • https://github.com/docker/docker/graphs/contributors	• 197 contributors • https://github.com/coreos/rkt/graphs/contributors	• 225 contributors • https://github.com/mesosphere/marathon/graphs/contributors
Community	• 37892 questions tagged • http://stackoverflow.com/questions/tagged/docker	• 42 questions tagged • http://stackoverflow.com/questions/tagged/rkt	• 476(?) questions tagged • http://stackoverflow.com/questions/tagged/marathon

Summary

Docker is more suitable for automation of applications distribution, it wins over VM's (which target machines automation rather than applications) and other modern containerisation approaches due to developer friendly architecture, tooling and widest community among alternatives.

Latest Thoughtworks tech radar placed docker into "Adopt" category: https://www.thoughtworks.com/radar/platforms/docker

References

- https://www.thoughtworks.com/radar/platforms/docker
- https://coreos.com/blog/rocket.html
- https://www.ubuntu.com/cloud/lxd
- https://coreos.com/rkt/
- https://mesosphere.com/blog/2016/09/30/dcos-universal-container-runtime/
- https://www.quora.com/Which-companies-use-Docker
- https://www.quora.com/Who-uses-CoreOS-in-production
- https://www.youtube.com/watch?v=2xSwDPBtt3E
- https://www.youtube.com/watch?v=UV3cw4QLJLs
- https://www.upguard.com/articles/docker-vs-lxc
- https://docs.docker.com/engine/understanding-docker/
- http://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-normal-virtual-machine