Introduction to Container Technology

Patrick Ladd
Technical Account Manager
April 13, 2016
Container Technology
Containers

- "Linux Containers" is a Linux kernel feature to contain a group of processes in an independent execution environment.

- Linux kernel provides an independent application execution environment for each container including:
  - Independent filesystem
  - Independent network interface and IP address.
  - Usage limit for memory and CPU time.

- Linux containers are realized with integrating many existing Linux features. There are multiple container management tools such as lxctools, libvirt and docker. They may use different parts of these features.
Container History

- 2000: JAILS ADDED TO FREEBSD
- 2001: LINUX-VSERVER PROJECT
- 2003: SELINUX ADDED TO LINUX MAINLINE
- 2005: FULL RELEASE OF SOLARIS ZONES
- 2006: GENERIC PROCESS CONTAINERS
- 2007: GPC RENAMED CONTROL GROUPS
- 2008: LINUX CONTAINER PROJECT (LXC)
- 2008: KERNEL AND USER NAMESPACES
- 2013: RED HAT ENTERPRISE LINUX
- 2013: DOTCLOUD BECOMES DOCKER
- 2014: GOOGLE KUBERNETES
- 2015: STANDARDS VIA OCI AND CNCF
- 2015: RHT CONTAINER PLATFORMS
- 2015: RHEL ATOMIC HOST
CONTAINERS ≠ VIRTUALIZATION
Underlying Technology

Enabling Technology in Linux has been present for many years

- Namespaces
  - Process
  - Network
  - Filesystem
  - User
  - IPC
  - UTS (UNIX Technology Services)
- cgroups - Control Groups
- Union (overlay) Filesystems
Namespaces
Process Namespaces

Original UNIX Process Tree
- First process is PID 1
- Process tree rooted at PID 1
- PIDs with appropriate privilege may inspect or kill other processes in the tree

Linux Namespaces
- Multiple, nested process trees
- Nested trees cannot see parent tree
- Process has multiple PIDs
  - One for each namespace it is a member of
Network Namespaces

Presents an entirely separate set of network interfaces to each namespace

- All interfaces including loopback are virtualized
- Ethernet bridges may be created
  - `ip link add name veth0 type veth peer name veth1 netns <pid>`
- Routing process in global namespace to route packets

Original Namespace:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
   link/ether 00:24:8c:a1:ac:e7 brd ff:ff:ff:ff:ff:ff

New Namespace:
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
Filesystem Namespaces

Clone / Replace list of mounted filesystems

- Similar to chroot
- Allows isolation of all mount points, not just root
- Attributes can be changed between namespaces (read only, for instance)
- Used properly, avoids exposing anything about underlying system
User Namespaces

Replace / Extend UID / GID

- Delete unneeded UID / GID from container
- Add / change UID / GID map inside container
- Use: root privilege in container, user privilege in base OS
IPC Namespaces

Similar to network namespaces

- Separate interprocess communications resources
- Sys V IPC
- POSIX messaging
UTS Namespaces

UTS : UNIX Technology Services

- Change inside container:
  - Hostname
  - Domain
Feature availability

- Filesystem separation
- Hostname separation
- IPC separation
- User (UID/GID) separation
- Processes stable separation
- Network separation
- Usage limit of CPU/Memory
- Mount namespace (kernel 2.4.19)
- UTS namespace (kernel 2.6.19)
- IPC namespace (kernel 2.6.19)
- User namespace (kernel 2.6.23 ～ kernel 3.8)
- PID namespace (kernel 2.6.24)
- Network Namespace (kernel 2.6.24)
- Control groups
Namespaces Summary

Isolation / Modification of Container processes from host

- PIDs
- Network
- Filesystems
- UID/GID
- IPC
- Hostname / Domain

See documentation on clone() system call for more complete details on functionality
(Warning: systems programmer jargon territory)
cgroups

- Way to allocate resources to processes running on a system
- Hierarchical and can be dynamically added, changed and removed
- Made up of several subsystems also called Resource Controllers

- Part of RHEL 6 & 7 Kernel
- Upstream since 2.6.24
- You must install userspace tools
  - Install libcgroup
CPU: cpu.shares=1024
CPUSet: cpuset.cpus=0-3
MEMORY: memory.limit_in_bytes=9223372036854775807
BLKIO: blkio.weight=1000

DB:
CPU=40% of 2 CPU
DISK=70%
cpu.shares=410
cpuset.cpus=2-3
blkio.weight=700

WEB:
CPU=50% of 4 CPU
DISK=20%
cpu.shares=512
cpuset.cpus=0-3
blkio.weight=200

TRANSFER:
CPU=10% of 1 CPU
DISK=10%
cpu.shares=102
cpuset.cpus=0
blkio.weight=100

EAP:
CPU=80% of WEB
cpu.shares=820

APACHE:
CPU=20% of WEB
MEMORY=4G MAX
cpu.shares=204
memory.limit_in_bytes=4G
Resource Controllers

- **blkio** — this subsystem sets limits on input/output access to and from block devices such as physical drives (disk, solid state, USB, etc.).
- **cpu** — this subsystem uses the scheduler to provide cgroup tasks access to the CPU.
- **cpuacct** — this subsystem generates automatic reports on CPU resources used by tasks in a cgroup.
- **cpuset** — this subsystem assigns individual CPUs (on a multicore system) and memory nodes to tasks in a cgroup.
- **devices** — this subsystem allows or denies access to devices by tasks in a cgroup.
- **freezer** — this subsystem suspends or resumes tasks in a cgroup.
- **memory** — this subsystem sets limits on memory use by tasks in a cgroup, and generates automatic reports on memory resources used by those tasks.
- **net_cls** — this subsystem tags network packets with a class identifier (classid) that allows the Linux traffic controller (tc) to identify packets originating from a particular cgroup task.
- **net_prio** — this subsystem provides a way to dynamically set the priority of network traffic per network interface.
- **ns** — the namespace subsystem.
Union (overlay) Filesystems
Union Filesystems

- Stacked / Layered Storage
- Copy on write
- Many available underlying implementations
  - Aufs
  - OverlayFS
  - btrfs
  - LVM
  - Device mapper
Container Security
CONTAINERS ARE NOT SECURE BY DEFAULT
Container Security

**ISOLATION OF HOSTS**
Host OS + SELinux maintained by trusted kernel engineers and frequently updated.

**ARE SOURCES TRUSTED?**
36% of Docker Hub official images contain high priority security vulnerabilities.*

**WHAT’S INSIDE CONTAINERS**
Red Hat + Black Duck = secure, trusted model for validating container contents.

**TRUST IS TEMPORAL**
New vulnerabilities are identified daily and containers become stale over time.

Container Isolation with SELinux

- **HOST OS**
  - **CONTAINER OS**
    - **APP**
    - **RUNTIME**
    - **CONTAINER**

- **SHARED SERVICE**
  - SELINUX
  - SELINUX
  - SELINUX
  - SELINUX
  - SELINUX

- **SELINUX**
  - SELINUX
  - SELINUX
  - SELINUX
  - SELINUX
  - SELINUX

**Diagram Shows:**
- Isolated Containers with SELinux providing security and isolation.
Red Hat Container Technology

- Stock Red Hat Enterprise Linux ("RHEL")
  - Full OS image
  - Docker packages added
  - All combinations of use
- Red Hat Enterprise Linux Atomic Host ("Atomic")
  - Stripped down OS image
  - Pre-installed docker packages
  - Only for container deployment
    - Limited additional packages
    - Different upgrade / update process (no yum)
  - Optimized settings for container deployment
  - Separate subscription from RHEL subscription
Atomic Formats

- Multiple environments available
  - Cloud image (qcow2)
  - RHEV (ova)
  - Hyper-V (vhd)
  - vSphere (ova)
  - Installer (iso)
Installing Atomic on kvm

• Create overlay of image
  
  ```
  qemu-img create -f qcow2 -o backing_file=rhel-atomic-cloud-7.2-12.x86_64.rhevm.qcow2 atomic-instance-0.qcow2
  ```

• Set up VM

• Customize VM startup
  
  • meta-data & user-data files
    
    • Host IP addresses
    
    • Login credentials

• Start VM
Register & Update Atomic

- Register Atomic
  - `subscription-manager register --username=myid`
  - `subscription-manager attach`
  - `subscription-manager list`

- Upgrade Atomic
  - `atomic host upgrade`

- Atomic upgrade status
  - `atomic host status`

- Recover from failed upgrade
  - `atomic host rollback`
Using Docker

• Getting help
  docker --help

• Information on docker install
  docker info
  docker network ls
Using Docker Images

- Download an image
  
  ```bash
docker pull rhel7:latest
  ```
- Modify Dockerfile
  - Update MAINTAINER
- Build image
  
  ```bash
docker build -t webserver .
  ```
- Show images
  
  ```bash
docker images
  ```
- Remove an image
  
  ```bash
docker rmi myimage
  ```
- Show all images
  
  ```bash
docker images -a
  ```
Using Containers

- **Start a container**
  
  `docker run -d -p 80:80 --name=myweb webserver`

- **Change content**

- **Start another container**
  
  `docker run -d -p 80:80 --name=myweb webserver`

- **List containers**
  
  `docker ps`

- **Stop container**
  
  `docker stop myimage`

- **Restart container**
  
  `docker restart myimage`

- **Remove container**
  
  `docker rm myimage`
Reference Materials

- Atomic documentation
- Atomic Download
  https://access.redhat.com/downloads/content/271/ver=/rhel---7/7.2.2-2/x86_64/product-software
THANK YOU