New Developments and Advanced Features in the Libvirt Management API

Daniel P. Berrangé <berrange@redhat.com>
What is Libvirt?

- C library API + language bindings
- Simple, stable, standard API
- Cross-platform, cross-hypervisor
- LGPLv2+ licensed
Stateless architecture
State-full architecture

Application

libvirt.so
- esx
- hyperv
- test
- remote

libvirtd
- qemu
- lxc
- uml
- libxl

RPC

QEMU
Disk access protection

• Danger scenarios:
  – 2 guests using same disk image
  – Same guest started twice on different hosts

• Disk access modes
  – Read-only, shared (<readonly/>)
  – Read-write, shared (<shareable/>)
  – Read-write, exclusive (default)
Sanlock

- Project from oVirt team
- Disk paxos algorithm
- Preferred use w/ SAN
- Discouraged use w/ shared filesystem (eg NFS)
- Manual leases
- Automatic leases
  - Indirect MD5(file path)
- Fence guests on lease failure
virtlockd

- Included with libvirt
- fcntl() based locks
- Requires use of shared filesystem
- Automatic leases
  - Direct file path
  - Indirect SHA256(file path)
  - Indirect LVM UUID
  - Indirect SCSI ID
virtlockd architecture

Application

libvirt.so
- esx
- hyperv
- test
- remote

RPC

libvirtd
- qemu
- lxc
- uml
- libxl

RPC

QEMU

virtlockd

filesystem
Fine grained access control

- Historic access read-write or read-only
  - eg Allow 'frank' to connect 'read-write'
- New ACLs on (object, subject, permission)
  - eg Allow 'frank' to 'start' guest 'apache'
- All libvirt public APIs
- All virt drivers in libvirtd (KVM, LXC, UML, etc)
- Pluggable backends (in-tree only)
Polkit ACLs

- 'polkit' is main (only) backend option
- Permissions mapped to actions
  - 'start' perm on 'domain' object
  - 'org.libvirt.api.domain.start'
- Object identifiers as properties
  - eg 'driver', 'id', 'uuid', 'name'
- Local UNIX users only
  - eg user 'fred' and group 'engineering'
Polkit rules

```javascript
polkit.addRule(function(action, subject) {
    if (action.id == "org.libvirt.api.domain.getattr" &&
        subject.user == "berrange") {
        if (action.lookup("connect_driver") == 'LXC' &&
            action.lookup("domain_name") == 'demo') {
            return polkit.Result.YES;
        } else {
            return polkit.Result.NO;
        }
    } else {
        return polkit.Result.NO;
    }
});
```
sVirt: SELinux

- Default: dynamic MCS w/ 'svirt_t' or 'svirt_tcg_t'
- Static label override per guest
- Base label override per guest
  - eg replace 'svirt_t' with 'my_svirt_t'
  - Still uses dynamic MCS
- Per disk label override
  - eg disable relabelling of shared CDROM
sVirt: DAC

- Default: fixed 'qemu:qemu' user/group
- Static label override per guest
- Dynamic or static image relabelling
Audit Logging

• Guest operations using host resources
  – All resources on start / stop
  – sVirt label assignment
  – vcpu hotplug
  – Memory balloon
  – Disk/net/pci/filesystem hotplug
  – cgroup properties ACLs
General logging

- Historic: syslog plain text
  - Formatted message
- New: systemd journald structured data
  - Raw message
  - Log priority
  - Log reason (debug/audit/trace/error)
  - Source file / line / function
Control Groups Defaults

- **Historic**
  - $ROOT/libvirt/{qemu,lxc}/\{guest-name\}

- **New non-systemd**
  - $ROOT/machine/{guest-name}.libvirt-{qemu,lxc}

- **New systemd**
  - $ROOT/machine.slice/{guest-name}.libvirt-{qemu,lxc}
Control Groups Custom

• Custom grouping

  <resource>
   <partition>/machine/production</partition>
  </resource>

• Non-systemd:
  – $ROOT/machine/production.partition

• Systemd
  – $ROOT/machine.slice/machine-production.slice
Tuning CPU

• Scheduler tunables
  – cpu_shares
  – \{vcpu,emulator\}_period
  – \{vcpu,emulator\}_quota

• CPU models
  – Named model
  – Host model
  – Host passthrough
Tuning Memory

- **NUMA policies**
  - Static CPU & memory placement via XML
  - Dynamic CPU & memory placement via numad
  - Guest NUMA topology

- **Allocation backing**
  - Huge pages, page sharing, locked

- **Memory tunables**
  - hard_limit, soft_limit, swap_limit, min_guarantee
Tuning Block

• Per guest tunables
  – weight
  – device_weight (per host block dev)

• Per guest disk tunables
  – \{total,read,write\}_iops_sec
  – \{total,read,write\}_bytes_sec
Tuning Network

• Per guest NIC tunables
  – QoS with 'tc'
    <bandwidth>
      <inbound average='1000' peak='5000' floor='200' burst='1024'/>
      <outbound average='128' peak='256' burst='256'/>
    </bandwidth>

• Migration tuning
  – MiB/second